68' MICRO JOURNAL

Australia A $4.75 New Zealand NZ $6.50
Singapore S $9.45 Hong Kong H $23.50
Malaysia M $9.45 Sweden 30-SEK

$2.95USA

Motorola VME-MACINTOSH-S 50 & Other 68XXX Systems

6809 68008 68000 68010 68020 68030

OS-9 The Magazine for Motorola CPU Devices FLEX SK*DOS

A User Contributor Journal

This Issue:

VOLUME IX ISSUE XII ● Devoted to the 68XXX User ● December 1987

The Grandfather of "DeskTop Publishing™"

000422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082

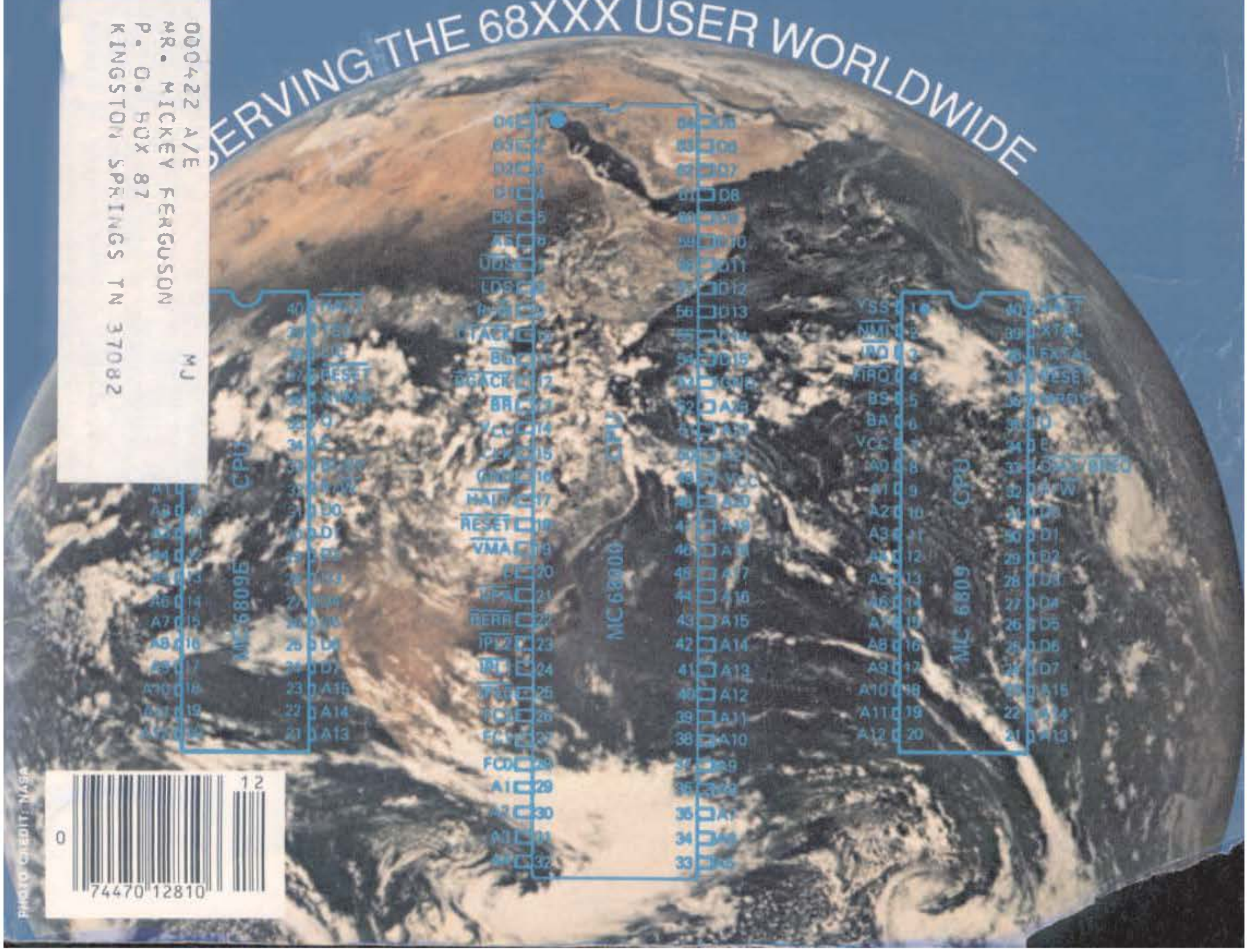
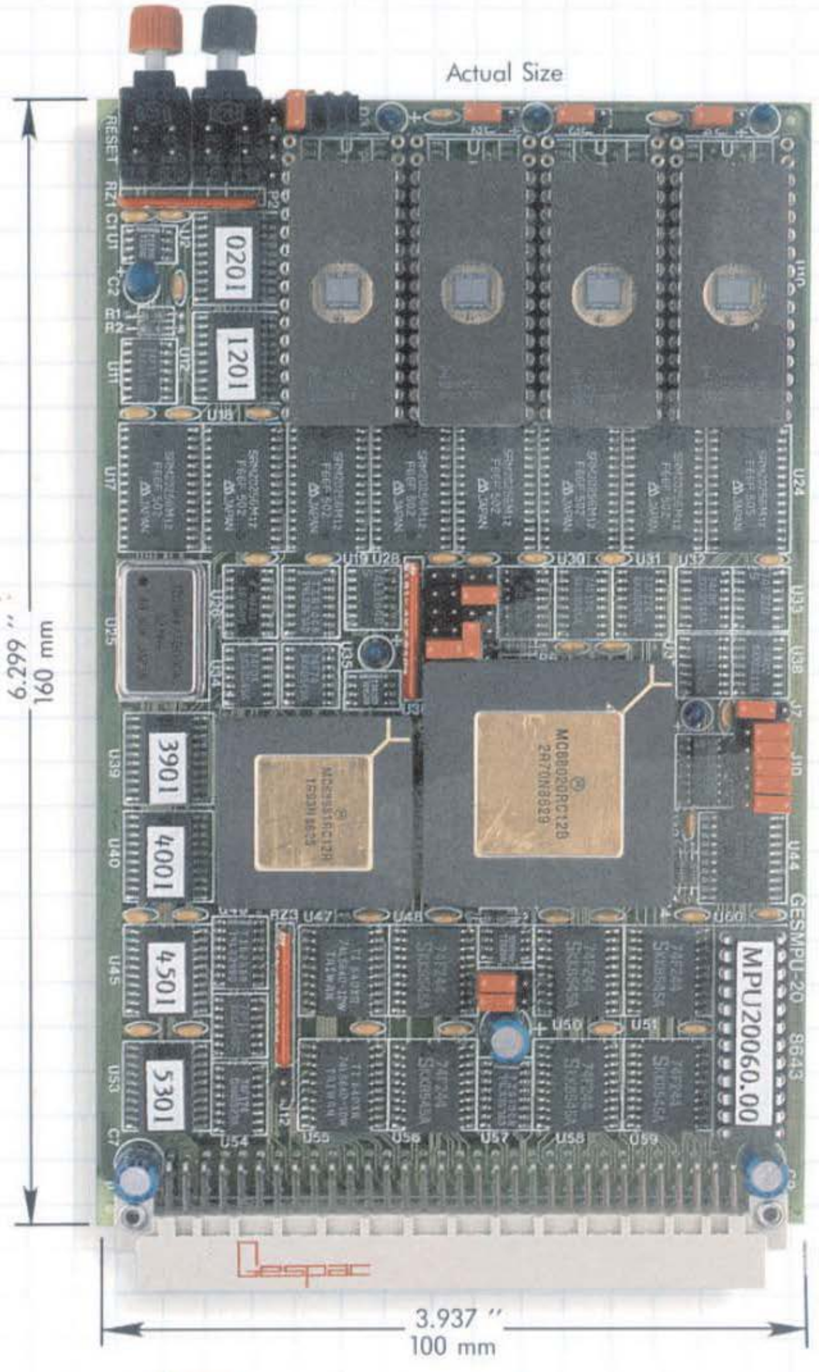


PHOTO CREDIT: NASA

GESPAC Gives You More Power Per Square Inch.



Here is the size, performance, and cost breakthrough you have been waiting for: The 68020 based GESMPU-20 from GESPAC.

You can now unleash an unprecedented amount of power into your applications. On just 25 square inches, we have squeezed a 12.5 MHz (16 MHz optional) 68020 32-bit microprocessor, a 68881 floating point coprocessor, 4 sockets for up to 512 Kilobytes of EPROM, and up to 512 Kilobytes of zero-wait-states CMOS RAM.

This board is totally expandable through its G-64 bus interface. And GESPAC has the largest variety of inexpensive memory, interfaces, controllers, and transducer cards anywhere. Plus real-time disk operating systems, high level languages, and other software tools. GESPAC has the total solution to your system integration needs.

Best of all, because our boards are small, they cost less. The new GESMPU-20 is priced below $1000 in one hundred piece quantity orders.

So why wait? Contact us today for information on the GESMPU-20 or any of the 150 G-64 bus system components from GESPAC—the leader in single Eurocard microcomputer products worldwide.

Call (602) 962-5559.

Gespac

IN USA - CANADA
50A West Hoover Ave.
Mesa, Arizona 85202
Tel. (602) 962-5559
Telex 386575

INTERNATIONAL
3, chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

THE GMX 020BUG DEBUGGER/DIAGNOSTIC PACKAGE

This extensive firmware package provides a broad range of program development tools and a complete suite of diagnostic programs for exercising GMX Micro-20 hardware.

The debugger includes commands for displaying and modifying registers and memory. If the optional 68881 Floating-Point Coprocessor is installed, its registers are also accessible. Memory can be displayed in hexadecimal and ASCII format, as floating-point values (single, double, extended or packed format), or as disassembled instructions (including FPC instructions). Memory modify can be done with hexadecimal values, with ASCII strings, with floating-point values, or with a one-line assembler which supports the full 68020 instruction set (although not the FPC instructions). Block move, fill, and search are also available.

Several different modes for tracing or executing user programs are provided, along with a powerful breakpoint facility. Programs and data may be downloaded from a host system or uploaded back to the host, and the GMX Micro-20 console may be used as a host system terminal. A serial printer may be hooked up, and used to make hardcopy listings of debugger sessions as desired.

The diagnostic firmware includes 90 test commands and 16 utilities. Complete test suites are provided for each functional block of the GMX Micro-20's hardware, including, for example, 9 different tests for memory, 9 tests for serial I/O ports, 2 tests for the 68881 FPC, and 9 tests for the optional memory management unit. For the peripheral control interfaces (floppy disk, SASI/SCSI hard disk or tape), test commands support a broad range of peripheral operations (read, write, format, etc.) so that the user may test both the interface and an attached device. Tests are provided for add-on I/O boards, including the ARCnet interface, I/O Channel interface, and parallel and serial expansion boards.

The utility commands allow the user to execute groups of test commands conveniently, repeat commands or command groups, enable or disable detailed fault reporting, count detected errors, or execute all the non-peripheral tests as a group. A switch option allows this last function to be invoked automatically at power-on or reset. Other utilities allow the user to check the state of the various jumpers and switches on the GMX Micro-20 directly.

In addition to the Diagnostic command package, 020Bug contains a confidence test which is always run after power-on or reset. This test does a quick checkout of the processor and the basic system elements that are needed for 020Bug operation. If any defect is found, an error code is signalled by on-and-off blinks of an LED.

DEBUGGING COMMANDS

MD	— Memory display
MM	— Memory modify
MS	— Memory set
BF	— Block fill
BM	— Block move
BS	— Block search
RD	— Register display
RM	— Register modify
OF	— Offset registers
BR	— Breakpoint set
NOBR	— Breakpoint delete
G	— Go to target code
GD	— Go, delete breakpoints
GN	— Go, stop after 1 instruction
GT	— Go, set temp breakpoint
T	— Trace
TC	— Trace on change of flow
TT	— Trace to temp breakpoint
LO	— Download
DU	— Upload
VE	— Verify download
TM	— Terminal mode
PA	— Printer attach
NOPA	— Remove printer
PF	— Port format
TD	— Time display
TS	— Time set
SD	— Switch directory
RS	— Restart system
OS	— Boot operating system

UTILITY COMMANDS

NV	— Non-verbose mode
SE	— Stop on error mode
LE	— Loop on error mode
LC	— Loop continual mode
ST	— Selftest mode
STL	— Selftest with LED mode
DE	— Display errors
ZE	— Zero errors
DP	— Display pass count
ZP	— Zero pass count
RL	— Read loop
WL	— Write loop
DJ	— Display baud rate jumper settings
DS	— Display switch
MJ	— Display MMU board jumper settings
SX	— Scan I/O expansion space

TEST COMMANDS

AN — ARCnet interface tests	
A	— Wakeup test
B	— DIP Switch test
C	— Interrupts test
D	— Buffer test
CA20 - On chip cache tests	
A	— Basic caching
B	— Unlike function codes
C	— Disable
D	— Clear
FD — Floppy disk tests	
A	— Set parameters
B	— Drive select toggle
C	— Side select toggle
D	— Restore
E	— Seek
F	— Format track
G	— Read
H	— Write
I	— Copy buffer
J	— Compare buffer
K	— Fill buffer
IC — I/O Channel tests	
A	— Print test pattern
B	— Bit rotate
MH — Miscellaneous hardware tests	
A	— 68881 FPC instructions
B	— 68881 FPC control functions
C	— Tick generator
D	— Interrupt sources
MT — Memory tests	
A	— Set function code
B	— Set start address
C	— Set end address
D	— Random inversion test
E	— March address test
F	— Walk-a-bit test
G	— Refresh test
H	— Random byte test
I	— Program test
J	— TAS test
K	— Test 0000-1FFF
L	— Partial longword writes test
MU — Memory Management tests	
A	— Map RAM data test
B	— Map RAM address test
C	— Map RAM partial write test
D	— Map RAM random data test
E	— Accessed bit reset test
F	— Address mapping test
G	— Accessed/Dirty bits test
H	— Valid/Write Enable test
I	— Task size test
PP — Parallel port tests	
A	— Print test pattern
B	— Continual test bit pattern
C	— Test bit pattern for 10 sec
PX — Parallel I/O expansion board tests	
A	— Data, handshake, and IRQ test
B	— P4 connector test
C	— Data and handshake toggle
SA — SASI/SCSI port with SASI device	
A	— Select drive
B	— Scan data lines
C	— Restore
D	— Seek
E	— Read
F	— Write
G	— Compare buffers
H	— Fill write buffer
I	— Test interrupt
J	— Park head
K	— Format
SC — SASI/SCSI port with SCSI device	
A	— Select drive
B	— Scan data lines
C	— Restore
D	— Seek
E	— Read
F	— Write
G	— Compare buffers
H	— Fill write buffer
I	— Test interrupt
J	— Stop drive
K	— Format
SI — Serial I/O tests	
A	— Select DUARTs
B	— Internal loopback
C	— External loopback
D	— Baud rates
E	— Parity modes
F	— Character lengths
G	— Handshake lines
I	— BREAK detect
J	— Interrupt output
K	— Continual handshake toggle
TA — Tape drive tests	
A	— Rewind
B	— Read
C	— Write
E	— Compare buffers
F	— Fill write buffer
G	— Erase

GMX 1337 W. 37th Place, Chicago, IL 60609
(312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Stacy Power

Contributing & Associate Editors

Ron Anderson	Dr. E.M. "Bud" Pass
Ron Voigts	Art Weller
Doug Lurie	Dr. Theo Elbert
Ed Law	& Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 Telex 510 600-6630

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit uilt 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage paid ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, e c. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

EXCITING SOFTWARE FROM THE LEADER...

microware

OS-9 ELECTRONIC MAIL

Flash your message on Electronic Mail Mail is a screen- or line-oriented program that runs on your OS-9/680X0 systems or over OS-9/NET. You can use distributed mailing lists or consecutive mailing list to get your message delivered. And received mail can be sent directly to your printer for immediate printout, spooled on a multiuser system or saved to a file. Mail features on-line help and complete, easy-to-understand documentation.
Electronic Mail $150.00.

PRINT SPOOLER

Spool it and Print it! Someone beat you to the printer? Don't blow your top while you cool your heels--get the OS-9/68000 Print Spooler and relax. The full featured Print Spooler automatically routes and monitors the status of your devices and the output files to be spooled. Now you can have a complete print spooling management system at an affordable price.
OS-9 Print Spooler $150.00

FORTRAN

Crunch It! with Our New FORTRAN 77 Compiler Now you have a powerful new tool to take full advantage of the 68000 family of microprocessors. With Microware's FORTRAN 77 Compiler you can generate code that uses system-wide modules instead of linking redundant copies of the standard library to each program. Result: less memory, less disk space, faster loading and external updating!
FORTRAN 77 Compiler $750.00

OS-9/ST

NEW for your Atari ST! Now you can have the power of OS-9 on your Atari 520 or 1040 ST. A true multi-tasking environment for professional real-time results. OS-9/ST is available in two configurations: Personal and Professional. Choose either version for true multi-user support... And all at a price that puts UNIX to shame.

Personal OS-9/ST combines the power of OS-9 with an interactive, structured Basic. **$150.00**

Professional OS-9/ST has a powerful Assembler, Linker and User Debugger and the tools to turn your Atari ST into a full C Language workstation. **$600.00**

OS-9/68020 C COMPILER

NEW "speed demon" C Compiler! Now you can get your hands on a highly optimized C Language power tool--the OS-9/68020 C Compiler. When coupled with the MC68881 math co-processor, this compiler will let you'll blast through complex math functions in the blink of an eye. All compiler/assembler/linker options are controlled by an intelligent compiler executive that spares you from memorizing compiler options and module-calling sequences. And the compiler includes library functions for memory management and system events, and much, much more! The new OS-9/68020 C Compiler is included with the Professional OS-9/68020 System Software Package.
New C Language Compiler $750.00

To order these exciting NEW products or for more information... ***CALL TODAY!***

Microware Systems Corporation
1900 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

West Coast Office
4401 Great American Parkway • Suite 220
Santa Clara, California 95054

Microware Japan. Ltd.
41-19 Honcho 4 Chome, Funabashi City • Chiba 273,
Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB	**Dr. Rudolf Keil, GmbH**	**Elsoft AG**	**Vivaway Ltd.**	**Microprocessor Consultants**	**Microdata Soft**
S:t Persgatan 7	Porphystrasse 15	Zelweg 12	36-38 John Street	92 Bynya Road	97 bis, rue de Colombes
Box 1309	D-6905 Schriesheim	CH-5405 Baden-Dattwil	Luton, Bedfordshire, LU1 2JE	Palm Beach 2108	92400 Courbevoie
S751-43 Uppsala	West Germany	Switzerland	United Kingdom	NSW Australia	France
Sweden	Phone (0 62 03) 67 41	Phone (056) 83-3377	Phone (0582) 423425	Phone 02-919-4917	Phone 1-768-80-80
Phone 018-138595	Telex 465025	Telex 828275	Telex 825115		Telex 615405
Telex 76129					

Microware is on the move. We have openings for Technical and Marketing Professionals.
Send your resume (in confidence) today and find out more about these exciting opportunities.

OS-9 and BASIC09 are trademarks of Microware and Motorola. UNIX is a trademark of Bell Laboratories, Inc.

MUSTANG -020 Super SBC ™

DATA-COMP proudly presents the first Under $5000 "SUPER MICRO".

The MUSTANG-020™

CPI
Mustang-020™
A DATA-COMP
Hi-Speed
Product

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact. 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk contr llers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super nucros". In practical applications it has numerous applications, ranging from scientific t education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process contr l, the MUSTANG-020 is the c st effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popular high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes c mmands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user pr grams.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP
Installed Systems World-Wide
OVER 18 YEARS OF DEDICATED QUALITY!

CPI
A Division of Computer Publishing, Inc.
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

Mustang-020 Mustang-08 Benchmarks

```
                                                      32 bit     Register
                                                      Integer    Long

IBM AT 7300 Xenix Sys 3                                 9.7
AT&T 7300 UNIX PC 68010                                 7.2        4.3
DEC VAX 11/780 UNIX Berkley 4.2                         3.6        3.2
DEC VAX 11/750        "       "                         5.1        3.2
68008 OS-9 68K 8 Mhz                                   18.0        9.0
68000 OS-9 68K 10 Mhz                                   6.5        4.0
MUSTANG-08 68008 OS-9 68K 10 Mhz                        9.8        6.3
MUSTANG-020 68020 OS-9 68K 16 Mhz                       2.2        0.88
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz                1.8        1.22

Main()
      {
              register long i;
              for (i=0; i < 999999; ++i);
      }
Estimated MIPS - MUSTANG-020 ..... 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs
```

OS-9

OS-9 Professional Ver	$850.00
*Includes C Compiler	
Basic09	300.00
C Compiler	500.00
68000 Disassembler (w/source add: $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz.	
Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232 requires 1 or 2 Adaptor Cards below.	335.00
RS232 Adapter Each card supports 4 additional ser. ports (total of 36 serial ports supported)	165.00
60 line Parallel I/O card Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group.	398.00
Prototype Board areas for both dip and PGA devices & a pre-wired memory area up to 512K DRAM.	75.00
SBC-AN Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call	475.00
LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

Special for complete MUSTANG-020™ system buyers - Sculptor $695.00. SAVE $300.00

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

THE PRO!

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

CPI
Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020

These *hi-speed* 68020 systems are presently working at *NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers,* worldwide, where speed, math crunching and multi-user, multi-tasking *UNIX C level V* compatability and low cost is a must!

Only the "PRO" version of OS-9 supported!

This is *HEAVY DUTY* Country!

MUSTANG-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5" -80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K Professional Ver. • Includes C Compiler ($500.00)		$850.00
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk controller		$395.00
Shipping USA UPS		$20.00
Total:	Save $1000.00	$5,299.80

complete sytem $4,299.80

UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$375.00
20 Mhz MC68020 Sys		$750.00

Note all 68881 chips work with 20 Mhz Sys

UPGRADES
Write or Call for Professional OS-9 "Full Bore" Upgrade Kit

NOTE: Only Professional OS-9 now available (68020 Version) Includes ($500.00) C Compiler 68020 & 68881 supported

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information.

complete
25 Mbyte HD System
$4299.80
85 Mbyte HD System
$5748.80

Data-Comp Division
CPI
A Decade of Quality Service
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 · Telex 510 600-6630 Hixson, Tn 37343

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

service For Over a Decade! 1,200 Titles •

A Division of CPI

S.E. Media
9900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU!** **All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO
PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

The C Programmers Reference Source. Always Right On Target!

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Lama Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter begins the discussion and presentation of a public-domain portable math library written in C by Fred Fish.

MATH LIBRARY

A math library is often taken for granted by many programmers. There are cases in which a math library is not available when one is needed. There are also cases in which a math library is available, but is too large, too slow, too inaccurate, or is otherwise somehow inappropriate for an application. Depending upon the C compiler and linker (if any), the entire math library may be included when only one function is called, wasting a large amount of memory space. Subtle problems with manufacturers' math libraries have been reported almost since they were initially released.

Fred Fish developed a portable math library in the C language for his own use several years ago and has placed it into the public domain. He had previously been using Fortran, which automatically provided the functions, for engineering classes, and was converting to the c language. He quickly encountered problems with portability, missing functions, compiler bugs, etc. In self-defense, he developed his portable math library, which he called "pml". His goals in developing it included portability and correctness first, with efficiency important but of lesser priority. Much of the source code, although not necessarily the object code, representing the portable math library, is devoted to debugging and testing the functions.

The portable math library depends upon the C compiler to provide a double precision floating-point data type, with associated elementary arithmetic operations for addition, subraction, multiplication, and division. With a loss of precision, plus changes to the pml.h header file and some of the library functions, single precision floating-point (float) data type could be substituted for double precision floating-point (double) data type. In any case, the portable math library routines assume that no underflow or overflow errors are trapped by the hardware or system software and attempt to avoid performing calculations which would elicit such actions. Rather it explicitly sets error indications internally, such that the application program has a standard method of intercepting, ignoring, or otherwise processing the error conditions, if desired.

As coded, the portable math library requires that the C compiler support the passing of structures to functions, in addition to the passing of pointers to structures to functions. With a loss of conciseness and elegance of algorithm statement, pointers to structures could be passed to functions, rather than the structures themselves.

The portable math library consists of a number of functions, representing the portable math library itself, two header files, several test driver programs, several test data files, and corresponding results files.

The portable math library is built on the following mathematical functions:

atan	- return arc tangent of argument
cos	- return cosine of argument
exp	- return natural exponential of argument
log	- return natural logarithm of argument
sin	- return sine of argument
sqrt	- return square root of argument

Mathematically speaking, cosine and sine are actually the same function, with a shifted argument. Defining two functions simplifies the implementation somewhat in that they may then recursively call each other to perform range reduction.

The following functions are the most highly machine-dependent in the portable math library, since they depend explicitly upon the format of the representation of the floating-point number:

dint	- return integer portion of argument
scale	- return argument scaled by power of 2
xexp	- return exponent of argument
xmant	- return mantissa of argument

Following is the pml.h header file. It provides definitions of debugging trace functions, which are called upon entry and exit of each function in the portable math library. It also provides definitions of many numerical constants.

```
/*
 * pml.h    this file is included in all of the
 *  portable math library routines when they are
compiled.
 * this is the place to put machine dependencies.
 *
 * it should be noted that for simplicity's sake, the
 *  parameters are defined as floating point
constants,
 *  rather than hexadecimal initializations of
allocated
 * storage areas.  thus the range of allowed numbers
 * may not exactly match the hardware's capabilities.
 * if the maximum positive double floating point
number
 * is exactly 1.11...e100 and the constant maxdouble
is
 * defined as 1.11e100, the numbers between 1.11e100
and
 * 1.11...e100 are considered to be undefined.  for
most
 * applications, this will cause no problems.
 */

#ifndef NO_DBUG
#include <dbug.h>
#else
#define DBUG_ENTER(a1)
#define DBUG_RETURN(a1)  return(a1)
#define DBUG_VOID_RETURN return
#define DBUG_EXECUTE(keyword,a1)
#define DBUG_2(keyword,format)
#define DBUG_3(keyword,format,a1)
#define DBUG_4(keyword,format,a1,a2)
#define DBUG_5(keyword,format,a1,a2,a3)
#define DBUG_PUSH(a1)
#define DBUG_POP()
#define DBUG_PROCESS(a1)
#define DBUG_FILE        (stderr)
#endif

#include <values.h>
#include <math.h>
#include <errno.h>

/*
 *  by one means or another, the following constants
 *  must be defined in one of the files listed above
 *  or in the code below; if a given C compiler does
 *  not provide one or both of the value.h and math.h
 *  files, the user must provide their equivalent.
 *
 *  MAXDOUBLE   =>  Maximum double precision number
 *  MINDOUBLE   =>  Minimum double precision number
 *  DMAXEXP     =>  Maximum exponent of a double
 *  DMINEXP     =>  Minimum exponent of a double
 */

#define MINDOUBLE        (1.0/MAXDOUBLE)
#define LOG2_MAXDOUBLE   (DMAXEXP + 1)
#define LOG2_MINDOUBLE   (DMINEXP - 1)
#define LOGE_MAXDOUBLE   (LOG2_MAXDOUBLE / LOG2E)
#define LOGE_MINDOUBLE   (LOG2_MINDOUBLE / LOG2E)

#define TANH_MAXARG      16
#define SQRT_MAXDOUBLE   1.304380e19

#define PI               3.14159265358979323846
#define TWOPI            (2.0 * PI)
#define HALFPI           (PI / 2.0)
#define FOURTHPI         (PI / 4.0)
#define SIXTHPI          (PI / 6.0)
#define LOG2E            1.4426950408889634074
#define LOG10E           0.4342944819032518276
#define SQRT2            1.4142135623730950488
#define SQRT3            1.7320508075688772935
#define LN2              0.6931471805599453094
#define LNSQRT2          0.3465735902799726547

/*
 *      IEEE DEPENDENCIES
 *
 *      cc -DIEEE => IEEE floating point format
 *
 */

#ifdef IEEE
#define X6_UNDERFLOWS (4.209340e-52)
 /* X**6 almost underflows */
#define X16_UNDERFLOWS (5.421010e-20)
 /* X**16 almost underflows */
#endif
```

```
#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif
#ifndef VOID
#define VOID void
#endif
```

Following is the pmluser.h header file. It defines the complex structure used in many of the portable math library functions.

```
/*
 * pmluser.h     include file for portable math
library
 * this file should be included in any compilation
 * module which accesses routines from the pml.
 */

typedef struct cmplx
{               /* Complex structure */
    double real; /* Real part */
    double imag; /* Imaginary part */
}   COMPLEX;
```

Following is the pmlerr.c file. It contains functions, structures, and values specifying the type of error processing to be performed for each type of error when corresponding error conditions are found.

```
/*
 *      This is a sample pml library error handler.
 *      It may be used as is, or another of the
 *      user's choice substituted.
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static struct pml_err
{
    int flags;    /* Flag word; bits defined in pml.h
*/
        char  *msg;        /*  Error  message
*/
    char *func;   /* Function in which error occured
*/
};

static struct pml_err pml_errs[] =
{
    CONTINUE | COUNT | LOG, "overflow", "exp",
    CONTINUE | COUNT | LOG, "underflow", "exp",
     CONTINUE | COUNT | LOG, "exponent overflow",
"scale",
     CONTINUE | COUNT | LOG, "negative argument",
"sqrt",
    CONTINUE | COUNT | LOG, "zero argument", "log",
     CONTINUE | COUNT | LOG, "negative argument",
"log",
    CONTINUE | COUNT | LOG, "argument > 1.0", "acos",
    CONTINUE | COUNT | LOG, "argument > 1.0", "asin",
    CONTINUE | COUNT | LOG, "overflow", "tan",
    CONTINUE | COUNT | LOG, "overflow", "cosh",
    CONTINUE | COUNT | LOG, "underflow", "cosh",
    CONTINUE | COUNT | LOG, "overflow", "sinh",
    CONTINUE | COUNT | LOG, "underflow", "sinh",
    CONTINUE | COUNT | LOG, "overflow", "asinh",
      CONTINUE | COUNT | LOG, "argument < 1.0",
"acosh",
    CONTINUE | COUNT | LOG, "overflow", "acosh",
     CONTINUE | COUNT | LOG, "argument >= 1.0",
"atanh",
    CONTINUE | COUNT | LOG, "underflow", "atan",
    CONTINUE | COUNT | LOG, "complex / 0", "cdiv",
    CONTINUE | COUNT | LOG, "complex 1 / 0", "crcp",
    CONTINUE | COUNT | LOG, "exponent underflow",
"scale",
    CONTINUE | COUNT | LOG, "no fractional part",
"dint",
};

static int err_count = 0;           /* error counter
*/
static int err_limit = MAX_ERRORS; /* pml error limit
*/

/*
 *      pmlcfs   Clear specified pml error handler
flags
 *      Clears the specified pml error handler flags
for the
 *      specified error.  Two or more flags may be
cleared
 *      by "or-ing" them in the call, as in "LOG |
CONTINUE".
 */

pmlcfs(err_code,flag_word)
register int err_code;
register int flag_word;
{
    if (err_code < 0 ||
        err_code > (sizeof(pml_errs)/sizeof(struct
pml_err)))
    {
        fprintf(stderr,"pmlcfs: invalid error code
%d\n",err_code);
    }
    else
    {
        pml_errs[err_code].flags &= ~flag_word;
    }
}

/*
 *      pmlcnt   get pml error count and reset it to
zero
 *      returns the total number of pml errors seen
 *      prior to the call, resets the error count to
zero.
 */

int pmlcnt()
{
    register int rtn_val;

    rtn_val = err_count;
    err_count = 0;
    return(rtn_val);
}

/*
 *      pmlerr   portable math library error handler
 *      provides a sample pml error handler.  Does
 *      not use any available "traps" so is machine
 *      independent.  Generally called internally by
the
 *      other pml routines.
 */

pmlerr(err_code)
register int err_code;
{
    register struct pml_err *err;

    if (err_code < 0 ||
        err_code > (sizeof(pml_errs)/sizeof(struct
pml_err)))
    {
        fprintf(stderr,
                "pmlerr:  invalid error code
%d\n",err_code);
    }
    else
    {
        err = &pml_errs[err_code];
        if (err->flags & LOG)
        {
            fprintf(stderr,"pml: %s in function
\"%s\"\n",
                err->msg,err->func);
        }
        if (err->flags & COUNT)
        {
            err_count++;
        }
        if ((err->flags & CONTINUE) &&
            (err_count <= err_limit))
        {
            return;
        }
        else
        {
            fprintf(stderr,"pml: error  limit
exceeded\n");
            fprintf(stderr,
                    "pml: task aborted with %d
error(s)\n",
                err_count);
            exit(-1);
        }
    }
}

/*
 *      pmllim   set portable math library error
limit
 *      sets the pml error limit to the specified
value
 *      and returns it to its previous value.
 *      Note that the default error limit is set
 *      at compile time by the value in "pml.h".
 */

int pmllim(limit)
register int limit;
{
    register int rtn_val;

    rtn_val = err_limit;
    err_limit = limit;
    return(rtn_val);
}

/*
 *      pmlsfs   set specified pml error handler
flags
 *      sets the specified pml error handler flags
for the
 *      specified error.  two or more flags may be
set
 *      by "or-ing" them in the call, as in "LOG |
CONTINUE".
 */

pmlsfs(err_code,flag_word)
register int err_code;
register int flag_word;
{
    if (err_code < 0 ||
        err_code > (sizeof(pml_errs)/sizeof(struct
pml_err)))
    {
        fprintf(stderr,
                "? pmlsfs --- invalid error code
%d.\n",err_code);
    }
    else
    {
        pml_errs[err_code].flags |= flag_word;
    }
}
```

The matherr.c function provides the error handling action. This routine should be replaced by the user's own error handler. The default action is to return zero. If the user wishes to supply the return value for the function, and to suppress default error handling by the function, the matherr function must return non-zero.

```
/*
 *        matherr     default math error handler
function
 */

#include <stdio.h>
#include "pmluser.h"
#include "pml.h"

static char    funcname[] = "matherr";

int    matherr (xcpt)
struct exception *xcpt;
{
    DBUG_ENTER (funcname);
    DBUG_RETURN (0);
}
```

The description of this portable math library continues in the next chapter with the presentation of the math functions.

C PROBLEM

Consider the problems involved in the process of certifying a new math library. The relatively large number of functions involved, each with its own unique characteristics, creates a complex situation. However, the certification of every function in a new math library is absolutely essential, because of the high probability of errors in uncertified functions.

Since the testing domain of a function of one variable is so large, exhaustive testing is impossible. Luckily, math functions are intended to be memoryless, so that, at least theoretically, the results of one call have no effects on the next independent call to the same or different function.

Although the only operations required of the underlying floating point software or hardware are the basic arithmetics of addition, subraction, multiplication, and division, the certification of these elementary operations is even more complex in some ways than the certification of the math functions, since each of the elementary operations involves two variables.

Suggest effective test procedures which will provide a degree of certainty that a math library is functioning correctly.

EXAMPLE C PROGRAMS

Following is this month's example C programs; they convert text files between unix and macintosh formats. Unix and related systems, such as xenix, ultrix, ctix, etc. (although not flex, os-9, and uniflex) use line-feed as a line terminator, whereas the macintosh uses carriage-return as a line terminator. In addition, the word processors on the macintosh insert control characters into even the ascii forms of the text files, and most macintosh word processors will not do word-wrap beyond a carriage return, so entire paragraphs should be treated as single lines.

```
/*  mac2unix.c - converts mac word to unix text */

#include <stdio.h>
#include <ctype.h>

main(argc,argv)
int argc;
char **argv;
{
    FILE *input = stdin, *output = stdout;
    int c;

    putc('\n', stderr);
    if (argc > 1)
    {
        if ((input = fopen(*++argv, "r")) == NULL)
        {
            fputs ("can't open input\n", stderr);
            exit (1);
        }
    }
    if (argc > 2)
    {
        if ((output = fopen(*++argv, "w")) == NULL)
        {
            fputs ("can't open output\n", stderr);
            exit (1);
        }
    }
    while ((c = getc(input)) != EOF)
    {
        switch (c)
        {
        case '\n':
        case '\r':
            putc('\n', output);
            break;
        case '\t':
            c = ' ';
        default:
            if (c >= 0x20)
                putc(c, output);
        }
    }
    if (input != stdin)
        fclose(input);
    if (output != stdout)
        fclose(output);
    exit (0);
}

/*  unix2mac.c - converts unix text to mac word */

#include <stdio.h>
#include <ctype.h>

main(argc,argv)
int argc;
char **argv;
{
    FILE *input = stdin, *output = stdout;
    char *p, string[256], string1[256];
    int a = 1, c = 1, d = 0, 1 = 0, pc = '\n';
    int ce = 0, nf = 0, sp = 0, zz = 0;

    putc('\n', stderr);
    if (argc > 1)
    {
        if ((input = fopen(*++argv, "r")) == NULL)
        {
            fputs ("can't open input\n", stderr);
            exit (1);
```

```
          }
    }
    if (argc > 2)
    {
        if ((output = fopen(*++argv, "w")) == NULL)
        {
            fputs ("can't open output\n", stderr);
            exit (1);
        }
    }
    while ((c = getc(input)) != EOF)
    {
        switch (c)
        {
        case '\\':
            pc = '\\';
            if ((c = getc(input)) == EOF)
                break;
            if (c == '\n')
                putc('\\', output);
            else
            {
                putc(l = c, output);
                break;
            }
        case '\n':
            if (l = ((pc == '\n') || nf || ce))
            {
                putc(l = '\n', output);
                if (ce)
                    --ce;
            }
            else
            {
                pc = c;
                if (((c = getc(input)) == EOF) ||
                    (c == '\n') || (c == '.'))
                    putc(l = '\n', output);
                else
                    putc(l = ' ', output);
                if (c != EOF)
                {
                    ungetc(c, input);
                    c = pc;
                }
            }
              break;
          case ':':
          case '.':
              if (pc != '\n')
              {
                  putc(l = c, output);
                  break;
              }
              if (fgets(string, 256, input))
              {
                      for (pc = zz = sp = ce = 0, p =
string;
                       (c = *p); ++p)
                  {
                      d = tolower(*(p + 1));
                      switch (c)
                      {
                      case '0':
                      case '1':
                      case '2':
                      case '3':
                      case '4':
                      case '5':
                      case '6':
                      case '7':
                      case '8':
                      case '9':
```

```
                          zz += (zz * 10) + (c - '0');
                          break;
                      case 'C':
                      case 'c':
                          if ((d == 'e') && (!pc))
                          {
                              ++pc;
                              ++ce;
                          }
                          break;
                      case 'F':
                      case 'f':
                          if ((d == 'i') && (!pc))
                          {
                              ++pc;
                              nf = 0;
                              *p += ('j' - 'f');
                              *(p + 1) += ('u' - 'i');
                          }
                          break;
                      case 'N':
                      case 'n':
                          if ((d == 'f') && (!pc))
                          {
                              ++pc;
                                nf = *(p + 1) += ('j' -
'f');
                          }
                          break;
                      case 'S':
                      case 's':
                          if ((d == 'p') && (!pc))
                          {
                              ++pc;
                              ++sp;
                          }
                          break;
                      }
                  }
                  if (!zz)
                      zz = 1;
                  if (ce)
                      ce = ++zz;
                  if (sp)
                      while (zz--)
                          putc('\n', output);
                  else
                      if (a)
                          putc('\n', output);
                      else
                      {
                          putc(',', output);
                          fputs(string, output);
                      }
                  pc = l = '\n';
              }
              c = '\n';
              break;
          default:
              if (c >= 0x20)
                  putc(l = c, output);
          }
          pc = c;
      }
      if (input != stdin)
          fclose(input);
      if (output != stdout)
          fclose(output);
      exit (0);
}
```

EOF

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL

LOOKING INTO PASCAL

If you have taken any courses in programming, you may have found that Pascal is listed as a requirement. Indeed, its original conception was a language for learning. Recently, I was looking into some graduate schools for computer science. Guess what language was in the entrance requirements. Yep! It was Pascal. In undergraduate courses, it is also used. I took a course in data structures and another in programming techniques. Both required Pascal.

Outside of academic concerns, Pascal has a place. I know of an engineering group that uses only Pascal in their projects. I recently went through the want ads and saw many programmer and software engineer jobs requiring a background in Pascal. It would be worthwhile to have some back ground in it.

I decided not to make this month's column a tutorial in Pascal. There are a number of fine books that can obtained from your local bookstore or library on the subject. Numerous articles have appeared in the 68' Micro Journal. And most colleges offercourses in it. You should have ample opportunity to learn it.

Rather this month I intend to go into some of the internal workings of it. We will take a look at how parameters are passed. We will examine the generation of an executable object code from P-Code. And some of the internal working of the code will be looked at.

It is beyond my capability to look at all of the Pascal compilers for OS-9. My compiler is by Microware. It is based on the ISO standard ISO7815.1, Level 0 language specification. There are some variations. At this point in time it would not be worthwhile to go into them. They do not make a large difference to this month's topic.

I have created the program for this month. It is in the listing at the end of the column. It is titled EQUATIONS. It will solve for simultaneous equations. The basic idea is that if you have N unknowns, then you must have N linear equations. The point where all the equations intersect is the solution. If the lines coincide then we have an infinite number of solutions. If the lines are parallel, there will no solution. We are only interested in the intersection.

By way of example, let us look at a simple problem. Given the following equations:

```
2X + 3Y = 8
 X +  Y = 3
```

We can probably guess at the solution, since this problem is relatively easy. But let us solve it by Cramer's rule. We solve for the determinent of the coefficient of the equation.

```
2  3
1  1  = 2*1 - 3*1 = -1
```

Here we multiply on the diagonal going right and subtracting the multiplication of the diagonal going left. As the matrix grows this process increases. Right multiplications are added and left multiplications are subtracted.

To solve for the the value of X we substitute the equations' results for the coefficients of X and solve again for the determinent.

```
8  3
3  1 = 8*1 - 3*3 = -1
```

We divide this by the first determinent, to find the value for X.

```
X = -1/-1 = 1
```

Similarly, we solve for Y.

```
Y = 2  8
    1  3 / -1 = ( 2*3 - 8*1 ) / -1 = 2
```

You can verify that these are the correct result. Choose either equation and try them. They will work.

This technique can be expanded to larger solutions, but the computations become more difficult. This is why I created this month's program. It will solve up to 20 simultaneous equations. It works on real numbers only. I have not put in any error checking. At worse the program will crash with Pascal Error #209 – Division by zero. Then we know we have found one of those cases we cannot solve for.

This should suffice for an explanation of what the program does and basically how the program works. As we go along a few of the finer details should become apparent. Our first stop in treating the Pascal source code in the Listing is to compile it. With the Microware compiler, I would enter:

```
OS9: pascal <equations
```

This line causes the compiler to input the listing and create a P-Code file. P-Code is a special file that has instructions understood by a P-Code interpreter. It is a special code that sits between the world of the source code and the object code. Its fits neither, but brings together both. It is not unique to the Microware Pascal, but is found in may other brands of the language.

One explanation for P-Code is that large programs would only fit on the "smaller" computers in this form. Large programs can be created and compiled to P-Code form. They can be run with an interpreter and support library. This process is analogous to Basic09 and its I-Code. Just like I-Code, P-Code is a form that can be loaded into memory. I-Code uses RUNB, the BASIC09 interpreter and library. P-Code uqsquses qPaqPascal qand SUPPORT, the Pascal interpreter and library.

There are other advantages to P-Code. It offers an opportunity to try and debug code. The process is much shorter to creating P-Code. Something can tried and changed, if necessary, with out much down time. The P-Code interpreter offers a more comprehensive error checking and diagnostic environment. If errors do occur, it is a shorter path to making corrections.

It would not be very worthwhile to dissect the P-Code in great depths. There are not many people that want to learn how to hand code it. The Pascal Compiler does a good job of it. But it may be useful to understand the basic framework of a P-Code file.

The file is created in 256 byte increments called sectors. The first one is the header. It contains information like the module name, program name, sync bytes ($87CD), type/language and attributes. The header is not exactly like the standard OS-9 header. Running IDENT or VERIFY will tell that the module header is incorrect.

The body of the file is made of sectors that contain code for the different programs parts or string constants. A program part or string constant always begins at the start of the sector. If it does not fill the sector completely, the remaining part is padded. For program parts, $3E is used and for string constant sectors, $00 is used.

The sector(s) contains the information about the program parts. It tells where the program and its string constants are located. It tells local and extended stack size requirements. The information is stored in a table. It is the same information that is printed at the end of a compiling session. When this months program is compiled, it should print something like:

```
PROC NAME        PSEC  PSIZE  LOCAL  STACK   CSEC  CSIZE
   0 EQUATION       5    429   4115   2016      7    272
   1 DETERM         3    111      9     19      4      0
   2 XRIGHT         1    129     11     17      2      0
   3 XLEFT          2    126     11     17      3      0
   4 XFER           4    177      8     14      5      0
                         972   4154   2083           272
```

Examining one of these will make things a bit more clear. DETERM is Procedure #1. It is located at PSEC (Program Sector) 3. The header is sector number 0. PSIZE (Program Size) is 111 bytes. The remaining bytes are padded. Local stack size is 9 bytes and extended stack size is 19 bytes. The CSEC (String constant sector) is at 4, but CSIZE (its size) is 0 bytes. So this is really a null sector. If we examined EQUATION, we see that the CSEC starts at sector 7 and is 272 bytes long. This means that it is stored in two sectors, the last being padded with $00. I think you can decipher the rest of these. They are not absolutely necessary to know, but may be useful.

Running the program is easy. Entering:

```
OS9:pascaln pcodef
```

will get the entire thing rolling. PASCALN is the Pascal interpreter. PCODEF is the file created by the compiler. It is a generic name. There are compile time options that can be used to change it.

Eventually, a program is created that can be translated into an object code. It has been debugged and is working fine. It is now ready to take the final step. It is ready to be turned into a executable module.

There are advantages to generating an executable object module out of the P-Code. The native code generated will run much faster, than it would with the interpreter. Although the support library is still needed, the interpreter can be dispensed with. No interpreter means more available memory. The executable module is reenterant, so more than one user can use it. The trade-off is the code generated is larger than the P-Code version and it takes some extra time.

To generate the assembler code from the P-Code file, PASCALT.PRUN is used. It is itself P-Code and must be run with the Pascal Swap interpreter, PASCALS. The following line will get the ball rolling.

```
OS9:pascals pascalt.prun s15k
```

This provides for a swap buffer of 15K bytes. This can be made larger, if necessary. But this will work for the intended use. The file PASCALT.MODL must also be in the commands directory. This will automatically be loaded.

The program will ask for the name of the P-Code file to be translated. Since I have let it default to PCO DEF, a simple return is sufficient. Next it will want the name of the output file. I entered EQUATIONS.ASM. Translate all procedures? The answer to this is 'Y'. It asked for the module's name. I entered, EQUATIONS. And finally, should line numbers be included. Again, I answered, 'Y'. With this all done it was on its way.

This process takes a bit of time. It is going through the P-Code file and using it to generate an assembly language listing. The code produced can be assembled with the Microware Interactive Assembler. Once finished the code can be viewed and editted, as any other assembly language source code.

There are a number of interesting items that can be learned from examining the assembly source code. The first item that attracts our attention is the Support routines. An attempt is made to use F$Link and find SUPPORT, the Pascal library. If it is successful the program continues. Otherwise F$Load is called. If it is not successful this time, F$Exit is called and it is all over. Assuming SUPPORT, the library for Pascal, is found, the returned value in the Y Register is the relative position of table of vectors pointing the Pascal routines.

The assembly source also uses PASCALDEFS, which is an equates file much like the OS9DEFS. It has a few OS9 Equates like F$LINK and F$Exit. It has File Control Block Equates for I/O. There are the Zero Page Equates for Pascal's working storage. And there is the SUPPORT routine offsets.

So when SUPPORT is found, the value in the Y register is saved and stored in SUPJTBL, the SUPPORT jump table address. Whenever a routine is needed to be used from the SUPPORT package, the value of SUPJTBL is loaded into the X Register and a JSR relative to X is made. Let's look at an example. Say we want to print a string, from the listing. The line:

```
writeln('Simultaneous Equations Solver by
Cramer's Rule');
```

translates to:

```
LEAX $A0D0,PCR
PSHS X
LDD #46
PSHS D
PSHS D
LDD 2,Y
PSHS D
LDX SUPJTBL
JSR WRTSTR,X
```

This part of the code puts the position of the string at $A0D0 and its length on the stack. (A few other things occur too. But they are incidental to our argument.) The last things are to load X with the table location and then jump to the subroutine WRTSTR relative to X.

The next item of interest is how a call is made to a function or procedure. Again it is best to take a look at an example. The line:

```
xfer(i);
```

in the main program is a good example. It is translated to:

```
LDD -6,Y
PSHS D
PSHS U
LBSR P4
```

This little sequence saves the value of variable 'i' onto the stack. The value of U, where the mainsdata area is located is pushed too. Its storage on the stack looks like this.

High Memory
value of variable 'i' (LSB)
value of variable 'i' (MSB)

address of main's data area (LSB)
address of main's data area (MSB)
return address (LSB)
return address (MSB)
Low Memory

This method is similar to the way other languages would pass parameters. The difference is that the main procedure's data area address is passed. This is true for any calling procedure.

On entry to the procedure XFER, we have to following code.

```
P4 LDA PROCN
LDB #4
STB PROCN
PSHS A
PSHS U
LEAU ,S
```

This sequence loads A with the calling procedure's number and save its in PROCN. A and U are pushed on the stack. U is loaded with the current position of the stack. This creates what is called the stack frame. It looks like:

High memory
Address of main's data area (2 bytes)
Return address (2 bytes)
Calling procedure number (1 byte)
Copy of U register (2 bytes)
Low memory

So now we have The U register pointing to the low memory area of the stack frame. Next the S register is adjusted to account for the local data area. So, the following shows the setup for XFER:

High Memory
Passed variables
Stack Frame
U Register
Local Variables
S Register
Low Memory

A few things do not appear here. If a FUNCTION were called and a returned value expected, this would be placed before the passed variables. Another thing is the Pascal Manuals recommends saving the value of DHOLD, from the direct page variables. So this should be pushed on the stack if you are planning to write you own assembly language code.

When the procedure is finished (in our case XFER), the process outlined here is reversed. Everything is returned to basically the way it was on entry to the routine. X is loaded with the return address and JMP ,X is executed.

Once the assembly code is satisfactory, the only thing left to do is to assemble it. This is done with:

```
OS9:asm equations.asm o=equations #12k
```

This will create an executable module in the commands directory. With the support package, it will run finding solutions to your simultaneous equations.

I think this should give you some food for thought. I realize that much of this somewhat technical. It may be beyond your interests. You can be a good Pascal programmer without having to be aware of what is happening. Also you may not be a Pascal programmer at all. And you think, why should I care about this?

There is a lot to be learned here. Knowledge of how things work can make you a great Pascal programmer. You will be able to go into the source and improve it. You'll be able to rewrite code and optimise its performance. You can create external modules that your programs can access.

There are some good techniques here for programming. If you are a assembly language programmer, you may find the methods of using the stack areas useful. Also, shown earlier was method to use external libraries.

That is it for this time. Have a good month!

LISTING

```
$TITLE Equations
{
   Name: Equations
   By:   Ron Voigts
   Date: 13-Aug-87
   Version: 1.00

   Function:
      This programs will evaluate 20 simultaneous
      equations and return the values of the
      variables associted with them.
}

{ t+ }

program equations(input,output);

CONST
   maxsize = 20;

TYPE
   matrix = ARRAY [1..maxsize,1..maxsize] of REAL;

VAR
    count : INTEGER;
    i,j   : INTEGER;
    a, m  : matrix;
    b     : array [1..maxsize] of REAL;
    dv    : REAL;

$PAGE
```

```
$SUBTITLE Determinent finder
{ This function returns the determinent of a matrix n X
n }
FUNCTION determ( a:matrix; n:INTEGER ):REAL;
VAR
     i  : INTEGER;
     d  : REAL;

{ A function to do the right cross multiply }
function xright( y:INTEGER ):REAL;
VAR
     d:REAL;
     i:INTEGER;
     k:INTEGER;

BEGIN
    d:=1.0;
    FOR i:=1 TO n DO
    BEGIN
        k:=y+i-1;
        if ( k > n )
            then k:=k-n;
        d:=d*a[k,i]
    END;
    xright:=d
END;

{ A function to do the left cross multiply }
function xleft( y:INTEGER ):REAL;
VAR
     d:REAL;
     i:INTEGER;
     k:INTEGER;

BEGIN
    d:=1.0;
    FOR i:=1 TO n DO
    BEGIN
        k:=y-i+1;
        if ( k < 1 )
            then k:=k+n;
        d:=d*a[k,i]
    END;
    xleft:=d
END;

BEGIN
    d:=0.0;
    IF (n = 1) THEN
        determ:=a[1,1]
    ELSE
        FOR i:=1 TO n DO
            d:=d + xright(i) - xleft(i);
    determ:=d
END;

$PAGE
$SUBTITLE Matrix Transfer
{ This procedure puts the m matrix into a
  and it adds b. }
procedure xfer(k:INTEGER);
VAR
    i,j   : INTEGER;

BEGIN
    FOR i:=1 TO count DO
        FOR j:=1 TO count DO
            a[i,j]:=m[i,j];
    FOR i:=1 TO count DO
        a[i,k]:=b[i]
END;
$PAGE
$SUBTITLE Main Program
BEGIN

     { Header for program }
     writeln;
     writeln('Simultaneous Equations Solver by Cramer''s
Rule');
     writeln('       Version 1.00 by Ron Voigts');

     { Get the number of equations }
     writeln;
     writeln('Enter the number of variables to be
found');
     writeln('Note: There must an equal number of
equations');
     readln(count);

     { Get the equations' values }
     FOR i:=1 TO count DO
     BEGIN
         writeln;
         writeln('Entering coefficients for equation
',i);
         FOR j:=1 to count DO
         BEGIN
            writeln('Enter coefficent #',j );
            readln(m[i,j])
         END;
         writeln('Enter the equations result');
         readln(b[i])
     END;

     { Find divisor }
     dv:=determ(m,count);

     { Get solutions to the equations }
     FOR i:=1 TO count DO
     BEGIN
         xfer(i);
         writeln('Variable #',i,' is ',determ(a,count)/
dv);
     END
END.
```

EOF

FOR THOSE WHO NEED TO KNOW **68 MICRO JOURNAL™**

Logically Speaking

The following is the beginning of a continuing series. Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

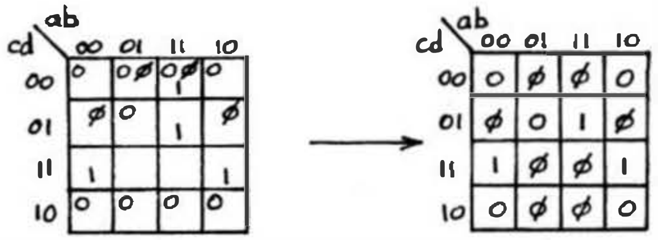
The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

LOGICALLY SPEAKING

Solutions to TEST FIVE

```
1. Let  a = barrier down      abc' + b'cd     T = 1
        b = photocell lit      d' + a'bc'd     T = 0
        c = car present      b'c'd + bc'd'     T = ø
        d = lamp OK
```

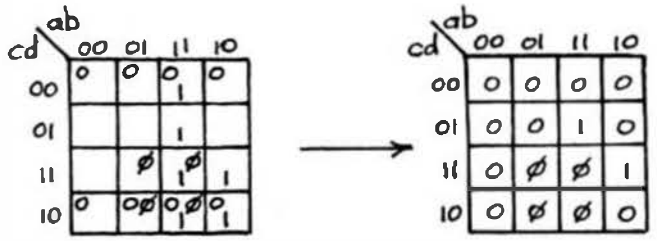


The K-map can be read out in four different ways here, depending on which phi's we choose to loop with the 1s. Check them all out for yourself! Maybe you can even find a fifth! The readings are :

(i) ab + cd (ii) ab + b'd (iii) ad + cd and (iv) ad + b'd

Of these, the last two are to be preferred as they can each be factored to a 3-literal expression. That is, (iii) d(a + c) and (iv) d(a + b').

```
2. Let  a = button                 ab + ac    T = 1
        b = Quarter                     d'    T = 0
        c = 2 dimes + 1 nickel          bc    T = ø
        d = bars in machine
```



The completed K-map reads out as abd + acd, which factors to ad(b + c).

BACK TO PUSSY-CATS FOR A WHILE

I know that by now lots of you are anxious to know wherein lies the logical flaw in our earlier CAT problem. The whole difficulty is really due to a peculiarity of the English language (and probably other languages too), where an expression can have different meanings, depending on what the speaker or writer actually intends to convey. For example, several years ago a certain laundry-detergent company advertised via TV "commercials" that

"NOTHING gets clothes cleaner than Blue XXX"

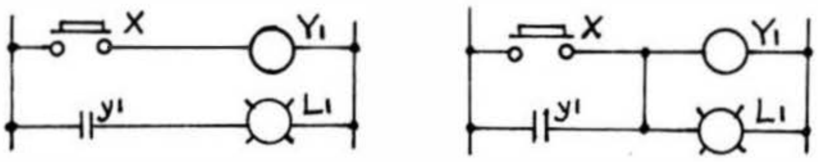
Sometimes this would be re-inforced with a smiling housewife assuring us tha^ "Nothing, ABSOLUTELY NOTHING, gets clothes cleaner!". What they meant to convey was that if you tested all the other detergents in the world, not one of them would wash clothes cleaner than their product. On the other hand, being the perverse sort of person that I am, I would interpret this as saying that if I used nothing-at-all my clothes would come out cleaner than if I used Blue XXX. Of course, if EVERYONE interpreted the message this way, the company concerned would find its sales plummetting, instead of rising, as a result of this ad.

In the same way, "no cat" in Statement 1 of our logic problem means that "if we gathered together all the cats in the world, not one of them would have 95 legs." Whereas "no cat" in Statement 2 means exactly the opposite. That is, instead of a gathering together of all cats in the world, it means "the complete absence of a cat." So, although "Uncle Fred" means EXACTLY the same thing throughout, in the case of Uncle Fred and Aunt Minnie, I guess I have to admit that "no cat" does NOT have the same meaning throughout our cat problem. So, although both statements are individually TRUE, they cannot be taken together to form a logical conclusion, as they are not talking about the same "no cat". It's as simple (or as complicated, whichever you like) as that!!

Mile 4, heading for Mile 5.

INTRODUCTION TO SEQUENTIAL CIRCUITS

Just as we did with combinational circuits (remember the difference between combinational and sequential?), we'll begin by analysing an already-designed circuit. The technique to be used will lead us directly to the actual design of sequential circuits, but first it's important to be able to examine an existing circuit, and to be able to say EXACTLY what this circuit is supposed to do.



(a) Diagram 13 (b)

So let's take a look at Diagram 13a which illustrates a very simple combinational circuit. We don't need any sophisticated method of analysis to be able to determine that it's combinational, because the state of L1 depends directly on the state of Push-Button X. That is, if X is not operated then L1 is not lit, and conversely, if X IS operated then L1 IS lit.

Now let's look at Diagram 13b, where we've simply connected together the coil of Relay-Y1 and the light L1. In so doing, however, we've changed our combinational circuit into one of the simplest of sequential circuits. It's action is fairly obvious. As it stands, we can see that it's at rest, as no current can flow either through X or y1. In the language of sequential-circuit theory we would say that the circuit is in a "stable state".

When X is operated, the coil of Y1 becomes instantaneously energised and L1 also becomes lit. The next thing that happens, possibly about 0.01 of a second later (and our technique HAS to take into account these tiny delays) is that the armature, or moving part, of the relay will activate and close NO-contact y1. At this time all action will cease, and the circut is said to be in stable state 2. Now observe the difference between 13b and 13a, in that, if we next release X, NOTHING HAPPENS TO EITHER THE RELAY OR THE LIGHT, because sustaining current continues to flow through the now-closed y1 contact. X has no further effect on the circuit, whether operated or not. Therefore this circuit is a sequential one, as the state of the output L1 does not now directly relate to the state of the input X. Initially it WAS so: X OFF meant L1 OFF, and X ON meant L1 ON, but after that things changed somewhat.

We're now going to re-define "Primary Control" slightly to mean one which causes a circuit to LEAVE a stable state. Make a good note of this new definition!!

Similarly, y1 is called a "Secondary Control" because its action results from an instability introduced into the circuit. In the case of 13b, the instability is due to the energisation of the relay-coil, as a result of which the armature MUST move mechanically, thereby closing its contact y1. This represents the circuit's attempt to find another stable state, which it may OR MAY NOT succeed in doing, as you'll see in future examples. In our example, however, it DOES succeed in finding a new stable state.

THE SEQUENTIAL FLOW-TABLE

The chart shown in Diagram 14 is known as a flow-table, and is used to both analyse and design sequential circuits. Note that it consists of 4 major squares (all that are necessary to analyse Diagram 13b mathematically), each of which is subdivided into 3 sections or Boxes, which I've labelled A, B and C. Box B is not used for analysis purposes, but plays an important role in the design process.

Our primary control, X, is tabulated along the top of the table, and y1 allocated to the rows. This is very important to remember -- don't EVER forget it. PRIMARY CONTROLS ARE ALLOCATED TO THE COLUMNS, AND CAUSE FLOW-ACTION TO MOVE HORIZONTALLY. SECONDARY CONTROLS ARE ALLOCATED TO THE ROWS, AND CAUSE FLOW-ACTION TO MOVE VERTICALLY. If you forget this, you're sunk!!

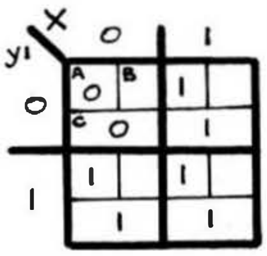


Diagram 14

Normally, I use red ink for the thin subdividing-lines, and also for entries in Box B. In fact, even though this is all printed in black ink, I shall later on refer to a "7", let's say, in Box A as a "black 7" and one in Box B as a "red 7", and I hope you too will stick to this convention in all your assignments. However, throughout this course, I shall merely emphasise the print of codes appearing in Section A, to make them "blacker" than those in Section B.

COMPLETING A FLOW-TABLE

Let's continue now, and use this table to find out what Diagram 13b actually does. We head the columns with 0 and 1, to signify X-Off and X-On, and the rows with 0 and 1, to signify contact y1 NOT operated, and contact y1 operated, respectively. OK, now let's go!! Into Box A will be written 0 or 1, depending on "the state of energisation of Relay Y1's coil". Note this. THE STATE OF ITS COIL, NOT ITS CONTACTS, which are not due to operate for a further split-second! So at the cross-point Xy1 = 00, we enter 0 in Box A, because under these conditions the relay's coil is NOT energised. In similar fashion, Box C will contain the state of energisation of all outputs (in this example only one), so we enter a 0 here too.

It is standard practice, commencing with a completely blank Flow-Table, to begin analysis in the top left-hand corner, where all controls are equal to 0.

Here is the first REALLY IMPORTANT thing to observe about our Flow-Table so far -- the fact that the code in Box A agrees with the co-ordinate to its left indicates that the circuit is in a stable state, and will remain there indefinitely unless someone does something about it. As the circuit itself is incapable of independent action, there is only one thing to be done. WE must operate X, thus moving horizontally to column 1, row 0, or in flow-chart language to Xy1 = 10. Looking once more at Diagram 13b, we note that if X is operated, both Y1 and L1 become energised, so we enter "1" in both Boxes A and C at this location.

And what do we see? Box A no longer agrees with the coordinate to the left, which means that the circuit is now unstable. Further, the entry of 1 in Box A FORCES the action to move vertically to Row 1 IN AN ATTEMPT TO FIND A STABLE STATE BY MAKING THE NUMBERS AGREE. So here we are now at Xy1 = 11 (interpretation, X operated, y1 operated) in our Flow-

Table, so we must look again at Diagram 13b to see what conditions now exist in the circuit, and make appropriate entries in the new Boxes A and C. We'll enter "1" in both spots to show that Y1 and L1 are energised still. Now we find that Box A DOES agree with the y1 coordinate "1" to its left, so our circuit is once more in a stable state where it will remain indefinitely.

So let's try releasing X to see what happens next! This action, of course, moves us horizontally to the left in our Flow-Table, to Xy1 = 01, where examination of the actual circuit-diagram results in entries of "1" in both Boxes A and C once more. Again, Box A agrees with the coordinate to its left, indicating a stable state. In fact, Box A throughout the whole of row 1 indicates a stable state, with no possibility of our ever moving out of this row. Unless, of course, we use a crowbar to physically pry the Relay apart, or more simply turn off the main source of power, in which case we'll pop back to Xy1 = 00, ready to begin the action all over again. Because our Row 1 contains codes in Box A which agree with the y1 coordinate, our circuit is quite content to stay in this stable state, and will not of itself make any effort to leave, so we are definitely stuck here unless WE choose to do something about it.

STABLE AND UNSTABLE STATES

Summarising then, an "unstable state" is recognised by the fact that Box A disagrees with the secondary-control state shown in the coordinate column to the left. Further, the circuit action will move vertically to the row whose number appears in Box A, in an attempt to make the numbers match. A "stable state" is one in which the numbers DO agree, and movement out of such a location can only take place horizontally. That is, via primary-control action.

ANOTHER ANALYSIS EXAMPLE

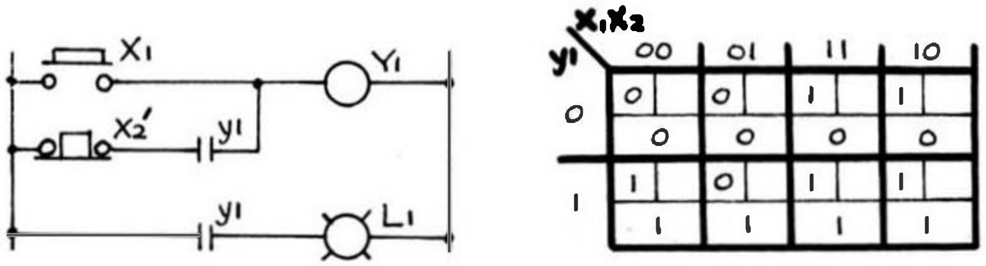


Diagram 15

The circuit of Diagram 15 has quite a few important differences from that of 13b, which we shall now examine on the Flow-Table. The top left-hand square is easy to complete, with a 0 being entered in both Boxes A and C. The circuit is stable (why?), which means that we'll have to operate a primary control in order to get something exciting to happen. Let's be methodical by moving into the next square to the right, ie, column 01, by operating X2. H'm, this doesn't seem to accomplish much, as Y1 and L1 will still remain OFF, so we enter 0s once more in the corresponding Boxes of this square. Box A tells us that we're still in a stable state, so let's try something else. Releasing X2 will add nothing to our knowledge, as we'll simply move back to our original square 000. So we'll try moving right once more into column 11, by operating X1 as well. The circuit diagram itself shows that under these conditions (that is, with both X1 and X2 operated) Y1's coil becomes energised. Note that L1 doesn't light -- not yet, anyway, as it has to wait for y1 contact to physically close, a short time later!!

Now we can enter 1 in Box A and 0 in Box C of location 110, and note that the circuit is now unstable (how do we know this?), and because Box A says "1" we MUST move to the corresponding row 1. In this row, y1 is equal to 1, which signifies that in the circuit itself the y1 contacts have now operated. The contact in series with X2' accomplishes nothing, as X2' is now open, but in row-3 of the diagram, L1 will energise, so we can enter 1 in both Boxes of location 111. Whereupon we see that that we are once more in a stable state, and again it's up to us what happens next.

We have 3 choices here. Namely, we can release X1 alone, or X2 alone, or we can release both simultaneously. Let's try the last choice -- that is, we'll jump from column 11 directly to column 00 in row-1 of our Table, and see what the network conditions will be with both buttons released BUT Y1'S CONTACTS STILL OPERATED (as they'll take a little while to drop out after a relay becomes de-energised). Evidently this move didn't change a thing! We're still in a stable state in row y1 = 1. Don't forget to complete the Boxes at each step!

Suppose we'd tried releasing X2 alone instead, what would have happened? Apparently the same thing, that is NOTHING! We would still be stuck in row y1 = 1. So, starting once more from location 111, let's release X1 alone, moving into location 011, and look again at the network. Note that with X1 unoperated, X2 and y1 operated, the coil of Y1 immediately gets de-energised. BUT THE LIGHT DOES NOT YET GO OUT! The introduction of 0 into Box A, however, tells us that at least we've succeeded in making the circuit unstable, and that it's going to move vertically to row 0. We find ourselves in an already completed location, in a stable state with both Y1 and L1 de-energised.

We now have only one location left to complete, so, starting from location 000, we'll slide right, by pressing X1 alone. So here we are in location 100, with X1 alone operated, and the circuit diagram tells us that under this condition Y1's coil becomes energised, BUT NOT L1 YET. Accordingly we enter 1 in Box A and 0 in Box C. Once again, we're unstable and therefore move vertically to row 1, where L1 comes on, and we're stable once more.

The Flow-Table is now complete, and from it we are able to follow the circuit action through any sequence of control changes we may choose to impose on it, without having to refer to the circuit-diagram any more to be able to say EXACTLY what is happening. The Table thus becomes a useful tool for diagnosing future problems should they occur in our little circuit.

NOTE : We're assuming at this stage of the game that all PRIMARY controls have "snap-action" and their contacts change state instantaneously, or at least a lot faster than do the secondary control contacts! More on this subject later.

A RELAY OSCILLATOR

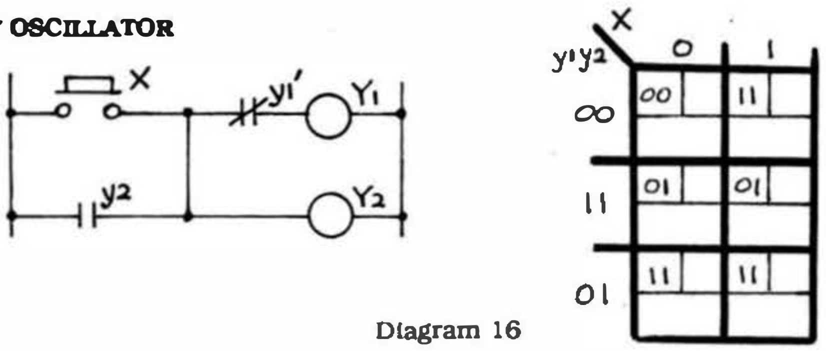


Diagram 16

The network we're going to analyse in Diagram 16 is unusual in that it has no outputs, and consists of just two relays and a push-button X. Normally, in circuits with more than one relay, we only draw up the first row of the Flow-Table to commence with (adding rows as we proceed with our analysis) and, of course, we label this first row with a 0 for each relay in the network. In this case it's labelled 00 for y1y2.

Commencing in location 000 then, we see that no current will flow in our circuit, and we therefore enter 00 in Box A to signify that both Relay coils are de-energised. This state, being a stable one, requires that we operate X in order for something to happen, thus moving to column 1 of the Flow-table. In the circuit itself we see that this action causes both relay coils to become energised, and we therefore enter 11 in Box A. As "11" does not agree with the coordinate "00", we're unstable and must move vertically to a row 11, which we'll open up immediately below our first row, and move our action down to location 111.

Referring to the circuit, we note that with X operated and both sets of relay contacts switched (this is what moving to row 11 means), only Y2 is energised, so we record this by entering 01 in Section A. And now what do we find? We are STILL in an unstable state, as 01 does not agree with the coordinate 11. So we must open up a row 01 into which to move vertically, and find ourselves now in location 101. The circuit shows us that under these condions (that is, both X and y2 operated, but y1 not) both relay coils are energised, necessitating a 11 entry in Box A.

Darn it, this is STILL an unstable state, sending us back up to row 11, where we just came from, which, of course, immediately shoots us back down again, and so on, and so on. So it appears we'll continue oscillating, with Y2 permanently energised and Y1 vibrating OFF and ON at high speed. Although we never reach a stable state, it's nevertheless apparent that as long as we hang on to X this oscillation will continue.

So let's try releasing X to see what happens! This could, of course, occur randomly either when we're in row 11 or in row 01. Let's assume we do it just as we arrive in row 11! This immediately slides us to the left into location 011. The circuit diagram tells us that this combination of controls causes Y2's coil alone to be energised, so we write 01 in Box A. This naturally sends us down to location 001, the corresponding network situation being that the coils of both relavs are energised, which we note in Box A.

The Flow-Table is now complete as far as anything we can do is concerned. So the Table tells us that, starting from an initial stable OFF state, the moment we press X the circuit is doomed to cycle continuously between y1y2 = 11 and y1y2 = 01, even though we release the button afterwards. As before, there is only one way to stop this, and that is to turn off the power supply.

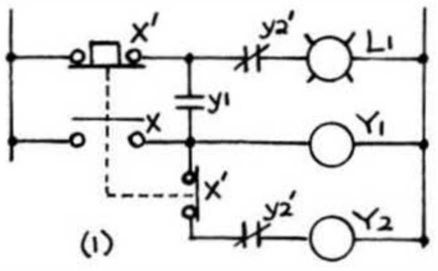
Note that no row y1y2 = 10 was developed in this Table. It therefore represents an "impossible" phi state. Examination of the circuit shows that it is in fact impossible for current to flow through Y1 and NOT through Y2 at the same time.

This concludes our analysis of simple sequential circuits, so once again we'll pause for a while to give you a chance to examine a few for yourself, so here goes with TEST SIX. Then, during the next stage of our journey, we'll discover, at last, how to design some beautiful sequential circuits of our own.

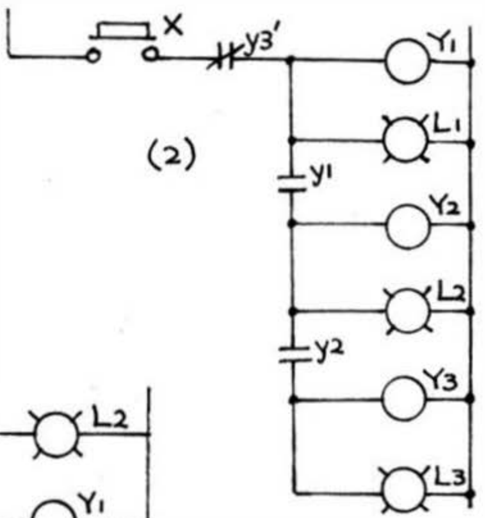
TEST SIX

Analyse the following circuits in terms of both relay and output action. That is, develop a Flow-Table for each.

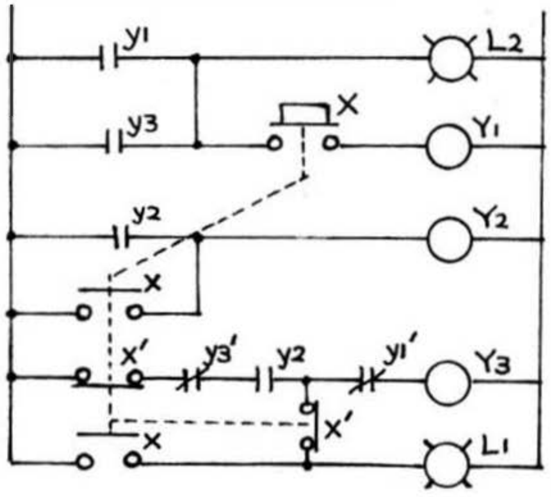
NOTE : In circuits showing several X-contacts linked by dotted lines (or even without these linkages, providing they each carry the same "name") it is understood that ALL such contacts operate simultaneously when the main X-button is operated or released.



(1)



(2)

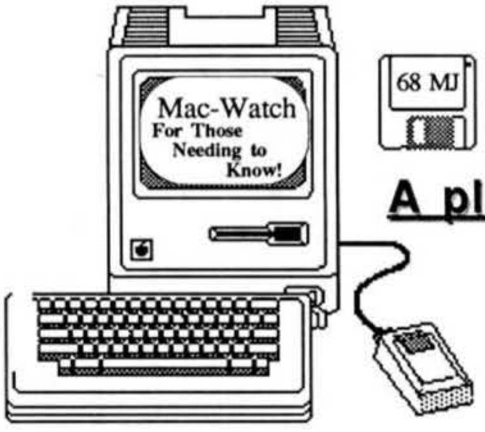


(3)

These examples certainly look MUCH more complicated than those I've been working on by way of demonstration, but if you take it one Flow-Table location at a time, you'll find that they are quite amenable to analysis, and you should not find it TOO difficult to complete the Tables. Who would have thought, just a few short lessons ago, that you'd be doing complicated stuff like this so soon? But that's progress!!

..... End of Mile 4, looking forward to an easy Mile 5.

EOF



The Macintosh™ Section

Reserved as a

A place for your thoughts

And ours.......

Mac-Watch

The Pain of Writing Macintosh Code or Finally a Way for the Rest of Us

Ever since Apple introduced the Macintosh series of microcomputers using the MC68000 CPU, I have used one for writing some of my articles and reviews. We received one of the very first, mainly because of our relationship with the Motorola series of 68XXX devices. However, unlike my usual practice with all our other systems, I have not felt prone to delve into the art of programming the Mac. As you who use or own one know, it is a "bear" to program!

If you have ever read, or even glanced at "Inside The Macintosh" you know what I mean. That book alone weighs at least 5 pounds and with the latest supplements is the most detailed documentation for any microcomputer I have ever seen! *And every bit of it needs to be understood to program the Mac.* Of the 25 or so chapters, it is said that one needs to know all the other 24 to understand any particular one chapter.

While the Mac is probably the most user friendly system available, it is, as I have indicated, also the most difficult to program. This primarily due to the graphics, windows and mouse capabilities and the rigid adherence required of the ROM "Toolbox" and OS routines that control every aspect of system functions. In all over 500 routines (some need never to be called by most programs, but must be available to other Toolbox and OS routines).

Practically everything passes through the Toolbox. As the Mac is a mouse and window system, even the most simple program requires an inordinate amount of detail programming, not to mention knowing the intricate details of most of those 500 or more Toolbox and OS calls (creating windows, placing them by pixel reference, calling the proper routines in the proper order, without neglecting any other needed routine(s), keeping up with where the mouse is, what is it doing (up or down, in or out of the box, etc.?) and a lot of other details not encountered in programming as most of us know it). Needless to say, for most users the thought of programming the Mac can raise more than a few chill bumps.

A Simple way to learn and write code, all at the same time

However, I discovered a way that is both simple and easy (even for me) and yet fun. And, the best part is, I learned what I need to know about most of those dreaded, but very necessary, Toolbox and OS routines while writing full Macintosh programs.

The answer lies in the use of several different applications listed below.

V.I.P. from Mainstay
5311B Derry Avenue
Agoura Hills, CA 91301

V.I.P. Translators to Lightspeed C and Pascal compilers. Also from Mainstay.

Also you will need either The **Lightspeed C or Pascal compiler**. Or both if you are really into programming. Lightspeed compilers are products of:

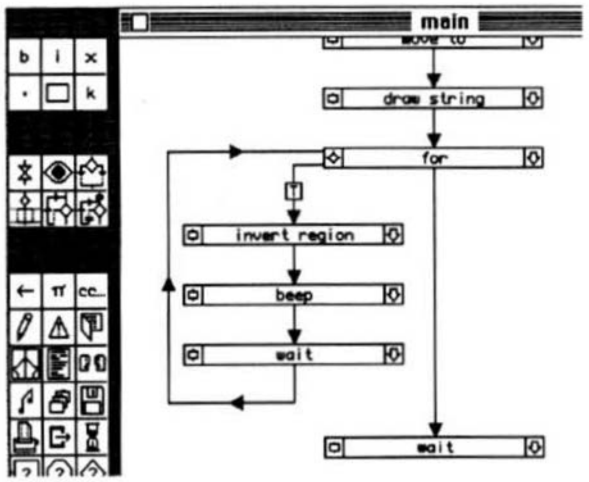
THINK Technologies, Inc.
135 South Road
Bedford, MA 01730

Because of the limitations of space I will not get deep into each product. This is not a review. as such, instead it is my way of licking a complex project and having fun while doing it. I will attempt to convey to you what each does in the scheme of developing worthwhile applications for the Macintosh, while not knowing anything about the infamous Toolbox or the OS routines. However, as you let V.I.P. actually do your programming for you (as I will explain below) you can very easily follow each and every move and call it executes. Thus gaining insight and experience about programming the Mac. As you use this method you will become progressively better at programming the Mac directly from most any available HLL, or even assembler, if that's your bag. Before you realize it, you are a full fledged Mac programmer. It is sorta like having a very experienced Mac programmer personally guiding you all the way...honest.

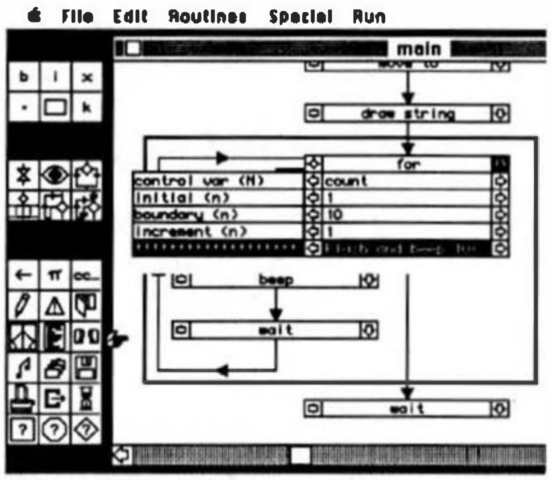
V.I.P.

In the beginning V.I.P. was the basic tool I used to get a program written as a *real* Mac application. While it's a very fine development tool in it's own right. V.I.P. is an interpretative system and runs like most BASIC programs, somewhat slow and requiring a "run-time" application should you convert the source to a semi-compiled project.

V.I.P. is an acronym for "Visual Interactive Programming™". You build a program by drawing a flow-chart type program outline and letting V.I.P. fill in all the myriad details. Major Objects, Logic Forms and Procedures are selected by icon clicking. Each in turn knows exactly which routines it needs to call. All you have to do is enter whatever constants, expressions or variables are required for your particular application. And each of these is prompted in such a manner you cannot easily make a fatal mistake.



Each procedure or logic form is indicated by an icon and each icon can be clicked open to tell V.I.P. what the arguments should be for each procedure or logic form called.



OBJECTS

Objects are shown in the icon box on the top left above. They are left to right and then down: BYTE - INTEGER - REAL - POINT - RECTANGLE - CONSTANT.

LOGIC FORMS

Logic Form icons are shown in the center box: ROUTINES - EXPANDED VIEW - IF-THEN-ELSE - SWITCH - WHILE-DO - FOR-NEXT.

TOOLBOX PROCEDURES

The Toolbox procedures are shown in the bottom left box: ASSIGN - MATH - COPY STRINGS - GRAPHICS - EVENTS - MENUS - WINDOWS - TEXT EDITING - DIALOGS - SOUND - RECORDS - I/O - PRINTING - BRANCHES - DATE/TIME. The bottom three (?) boxes are reserved for future expansion.

From this it can be seen that all the major functional programming problems are addressed by V.I.P. In all, over 200 major Toolbox and OS procedures are directly called by V.I.P. and remember, many of the others are never called by the programmer directly anyway.

The Process

Here is how I get the binary object code application programmed and running like any other Mac application, complete with full mouse and window support.

First, I build and debug the program in V.I.P. Experience was gained from the many sample programs included with V.I.P., until finally I found myself (several weeks, couple of hours a night, four or five nights a week) able to write fairly complex Macintosh applications and understand what they required of the Macintosh environment.

As I wrote and debugged each in V.I.P., I then used the V.I.P. to Lightspeed C or Pascal Translators to translate my V.I.P. code to C or Pascal source code. From then on it was a simple matter to call the Lightspeed C or Pascal compiler, load in the necessary libraries, translated V.I.P. source code and compile to object code as any other C or Pascal source program. *I might add that the V.I.P. translators only works with the Lightspeed compilers.*

The V.I.P. translators are very efficient, considering the differences involved. The code developed compared favorably with code generated directly from the Lightspeed compilers. And both the V.I.P. programs and the Lightspeed compilers allow complete print formatting (pretty printing). I learned a lot by printing out the code from each operation and then comparing one against the other.

I found within several weeks I was beginning to write code directly in the compiler's editor without need of the assistance of V.I.P. Or more aptly put - "I graduated to be a "real" Macintosh programmer in a period of several months." *And not the least of which, it was actually fun!*

The Lightspeed Compilers

The compilers are excellent tools for program development. They are exceedingly fast and are furnished complete with all necessary library and include files, even including the newer Mac II and SE. The editors supplied with each are a joy to use and have many features not found with other compilers.

With the early assistance of V.I.P. I was able to progress to the point of developing almost anything I wanted directly in the compiler editor and then using the compilers run and debugging functions to compile a running application. However, I still go back to V.I.P. if time is essential.

We have every Macintosh development package ever offered for the Macintosh. And almost everything available from the Developers Association for developing Mac software. MPW Macintosh Programmers Workshop, MDS Macintosh Development System, all MPW Utilities, all the compilers and assemblers, MacApp and lots more.

Yet, I preferred the process outlined above to any of the "official" stuff, including MacApp (Object Pascal with most all the window, mouse, menu and other normal routines already coded), which was until I received the V.I.P. systems and the Lightspeed compilers, the easiest way of developing Mac applications.

Recently we received the latest in "easy" Mac programming tools and applications - "HyperCard". While HyperCard is simple to use, to program it properly and to it's full potential, you must buy the reference manual (not included by Apple) and then learn another programming language - HyperTalk", not difficult but certainly different. Even then the results are restrictive and do not allow full access to all the Mac potential. I feel it is easier, and far more efficient, to go the way outlined above, rather than the HyperCard/HyperTalk method. When you get your application running properly, you will have done it your way, with all the latest bells and whistles available.

For those of you who have wanted to do more with your Mac but dreaded the extensive research required to even begin, or decided that Hyper-- was not sufficient for your projects, well, now there is a way, **V.I.P., V.I.P. Translators and Lightspeed C or Pascal compilers.**

One final note. If you are considering a compiler only, either C or Pascal, *I can honestly say the Lightspeed systems for the Mac are the fastest and best I have used yet.* And if you don't mind things slightly slower than pure object code but faster than BASIC, then V.I.P. as a standalone system is right down your alley. Either way - ***Now You Can Be A Macintosh Programmer!***

DMW

Just a little more:

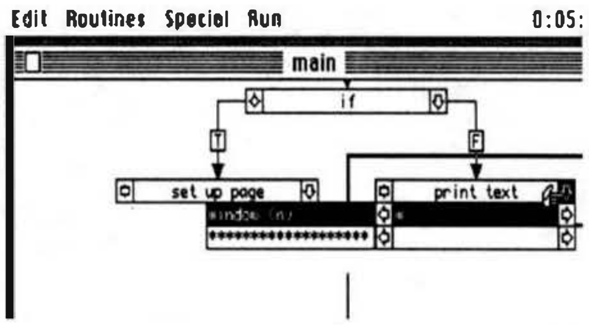
After finishing the above article, I felt a little gnawing something that kept telling me I had goofed somehow this time. So I let it "jell" for a day or so and went back and looked it over again, with my "honest Abe", critical eye. It was then I realized what it was that was nibbling away at my thoughts of reporting on these programs...*it just appears too simple!* I am so accustomed to allowing V.I.P. do my thinking for me, I assumed that it would be super clear to everyone else. So, a little more, in order that you might see for yourself what a tremendous aid and help a suite of development programs such as these can do for the beginning (and experienced) Macintosh programmer.

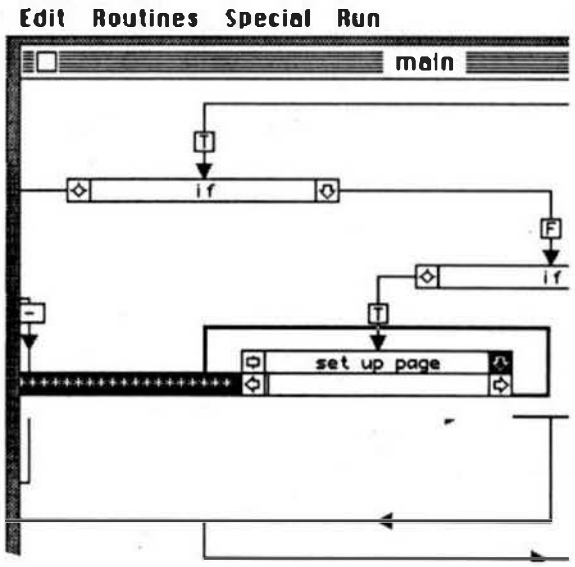
The first graphic below is, for example, what the entire page set-up and printer calling routines look like in V.I.P.. Two simple procedure boxes.

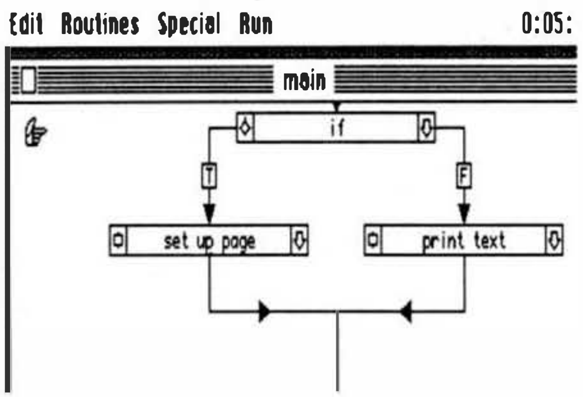
The second and third graphics show what information is required to be entered to have the printer option available in the final program.

Finally a listing of a typical page set-up and printer calling routine in Macintosh Lightspeed Pascal. While the code may not be exactly what would be output by the direct translation of the graphics shown. It does indicate the approximate amount of code that must be written, in source Pascal, to achieve a printing function in a final compiled object program. If you note the second and third graphics, only **one** entry has to be entered by the V.I.P. programmer. The line with asterisks is for entering a comment. I guess that tells the whole story a lot better than anything else I could write. Now, I believe I have made the point I wanted to make at the beginning.

DMW







```
Boolean PrinterOK = false;

static  THPrint hPrint;
static  TPPrPort pPrPort;

InitPrinter ()
{
        PrOpen ();

        if (PrError ())
                return;

        PrinterOK = true;

        PrintDefault (hPrint = (THPrint) NewHandle (sizeof (TPrint)));
}

DoPageSetup ()
{
        InitCursor ();
        PrStlDialog (hPrint);
}

static  Boolean Aborted;

Boolean AbortPrint ()
{
        EventRecord Event;

        if (Aborted)
                return (true);

        if (PrError ())
        {
                Aborted = true;
                return (true);
        }

        if (EventAvail (keyDownMask, &Event))
```

```
                if ((Event.modifiers & cmdKey) && ((unsigned char) Event.message == '.'))
                {
                        Aborted = true;
                        return (true);
                }

        return (false);
}

PrintWindow (WP)
        register    WindInfoPtr WP;
{
        register    PEHandle hPE;
        Rect PageRect;
        int     LinesPerPage,
                NumPages,
                OldScrollVal;

        PERemoveGap (WP -> hPE);

        SetPort (pPrPort);
        TextFont (FontNum);
        TextSize (FontSize);
        TextFace (0);
        TextMode (srcOr);

        CopyRect (&(**hPrint).prInfo.rPage, &PageRect);
        hPE = PENew (&PageRect);
        PESetHText ((**WP -> hPE).hText, hPE);
        (**hPE).tabWidth = (**WP -> hPE).tabWidth;

        LinesPerPage = (PageRect.bottom - PageRect.top) / (**hPE).lineHeight;
        NumPages = ((**hPE).nLines + LinesPerPage - 1) / LinesPerPage;

        if (AbortPrint ())
                goto EndPrint;

        OldScrollVal = GetCtlValue (WP -> VScroll);

        if ((*(char *) 0x947) == -bDevLaser)
        {
                TPPrint pPrint = *hPrint;
                if (pPrint -> prJob.iFstPage == 1)
                {
                        NumPages = min (NumPages, pPrint -> prJob.iLstPage);
                        goto LaserWriter;
                }
        }

        SetCtlValue (WP -> VScroll, 0);
        PrOpenPage (pPrPort, nil);
        if (!AbortPrint ())
                PEUpdate (hPE);
        PrClosePage (pPrPort);

        for (; --NumPages;)
        {
                if (AbortPrint ())
                        break;
                SetCtlValue (WP -> VScroll, GetCtlValue (WP -> VScroll) + LinesPerPage);
                PrOpenPage (pPrPort, nil);
                if (!AbortPrint ())
                        PEScroll (0, (long) - LinesPerPage, hPE);

                         PrClosePage (pPrPort);
                 }

                 goto EndPrint;

 LaserWriter:

                 SetCtlValue (WP -> VScroll, LinesPerPage * (NumPages - 1));
                 PrOpenPage (pPrPort, nil);
                 if (!AbortPrint ())
                         if (NumPages == 1)
                                 PEUpdate (hPE);
                         else
```

```
                                PEScroll (0, (long) LinesPerPage * (1 - NumPages), hPE);
                PrClosePage (pPrPort);

                for (; --NumPages;)
                {
                        if (AbortPrint ())
                                break;
                        SetCtlValue (WP -> VScroll, GetCtlValue (WP -> VScroll) - LinesPerPage);
                        PrOpenPage (pPrPort, nil);
                        if (!AbortPrint ())
                                PEScroll (0, (long) LinesPerPage, hPE);
                        PrClosePage (pPrPort);
                }

EndPrint:

                SetCtlValue (WP -> VScroll, OldScrollVal);
                DisposHandle (hPE);
                SetPort (WP);
        }

        DoPrint ()
        {
                WindInfoPtr WP = (WindInfoPtr) FrontWindow ();
                TPrStatus prStatus;
                DialogPtr DP;
                int     Copies;

                InitCursor ();

                if (!PrJobDialog (hPrint))
                        return;

                UpdateAllWindows ();

                Copies = (**hPrint).prJob.iCopies;

                if ((**hPrint).prJob.bJDocLoop == bSpoolLoop)
                        Copies = 1;
                else
                        (**hPrint).prJob.iCopies = 1;

                DP = GetNewDialog (PrintDlog, nil, -1L);
                DrawDialog (DP);

                Aborted = false;

                pPrPort = PrOpenDoc (hPrint, nil, nil);

                for (; Copies--;)
                {
                        if (AbortPrint ())
                                break;
                        PrintWindow (WP);
                }

                PrCloseDoc (pPrPort);

                if (!AbortPrint ())

                        if ((**hPrint).prJob.bJDocLoop == bSpoolLoop)
                                PrPicFile (hPrint, nil, nil, nil, &prStatus);

                DisposDialog (DP);
        }
```

EOF

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTWARE

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE

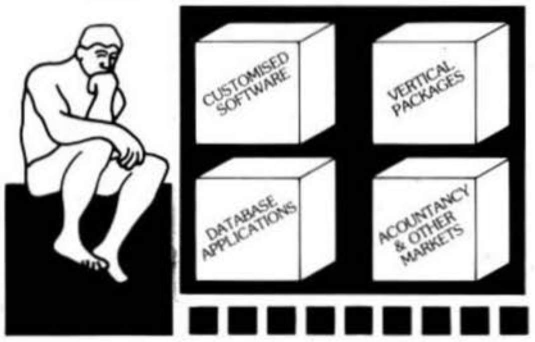
With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost,

Facts **Features**

Sculptor for 68020
OS-9 & UniFLEX
$995



DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Packed, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

–	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
–	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type

MUSTANG-020 Users - Ask For Your Special Discount!

* Tandy CoCo III Special - Reg. $595 * Special $389 *

	*	**	***		*	**	***
MUSTANG-020	$995	$199	$595	PC/XT/AT MSDOS	$595	$119	$595
OS/9 UniFLEX 6809	"	"	"	AT&T 3B1 UNIX	"	"	"
IBM Compatibles	"	"	"	SWTPC 68010 UniF	$1595	$319	$797
Tandy CoCo III	Special $389.00			SWTPC 68010 UNIX	$1990	$398	$995

... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

* Full Developement Package
** Run Time Only
*** C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex: 5106006630

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00

DYNAMITE+ – Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.
CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95
F, S, " " $100.00 - O, object only $150.00
U, " " $300.00

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.
F, S, CCF - $198.00

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.
*FLEX, SK*DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers.* "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*
F, S and CCF - $195.00

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer* OS-9 with many new commands and sub-fimctions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.
CoCo OS-9 $39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK*DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*
F, S and CCF - $295.00

C Compiler from Introl – Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F, S and CCF 5" - $190.00 F, S 8" - $205.00

PASCAL Compiler from OmegaSoft (now Certified Software) -- For the *PROFESSIONAL*; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible.
OS-9, F, S and CCF - $550.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA – A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*
*FLEX, SK*DOS, CCF - $99.95*

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
Color Computer ONLY - $58.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. · Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.
FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.
Like compilers for other languages, FORTHBUILDER can operate in "batch mode".
The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.
FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.
F, CCF, S - $99.95

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems
FOR 6809 FLEX-SK*DOS(5/8")
Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!
XDMS-IV Data Management System
XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.
POWERFUL COMMANDS!
XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriente which allows almost instant implementation of a process design.

SESSION ORIENTED!
XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV
IT'S EASY TO USE!
XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)
The possibilities are unlimited...
FOR 6809 FLEX-SK*DOS(5/8") $249.95

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, S, CCF - $99.95
Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.
Special -- CCF $35.00; F, S $50.00
OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. GeneOrate OS-9 Memory modules under FLEX, SK*DOS.
*FLEX, SK*DOS, CCF, OS-9 $99.00*
Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, S, CCF $150.00
MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
F, S, CCF - $75.00
XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
F, S, CCF - $98.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. · Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

UTILITIES

Basic09 XRef from S.E. Media – This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
O & CCO obj. only - $89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF – produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE – Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view t e overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
F, S, CCF --- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media – A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
U - $219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. BASIC TOOL-CHEST $29.95
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. FLEX, SK*DOS UTILITIES KIT $39.99
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: s ow sector order in free chain
   XL.CMD: super text lister
3. ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. WORD - PROCESSOR SUPPORT UTILITIES $49.95
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. UTILITIES FOR INDEXING $49.95
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK -- Double peek
3. DPOKE – Double poke
4. FPOS -- Current file position
5. FSIZE – File size
6. FTRIM – removes leading spaces from a string
7. GETPR -- returns the current process ID
8. GETOPT -- gets 32 byte option section
9. GETUSR – gets t e user ID
10. GTIME – gets the time
11. INSERT – insert a string into another
12. LOWER -- converts a string into lowercase
13. READY -- Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR – changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE – adds spaces to a string
19. SWAP -- swaps any two variables
20. SYSCALL -- system call
21. UPPER -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
See Review in January 1987 issue of 68 Micro Journal

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.
CONFIG one time system configuration.
CHANGE changes words, characters, etc. globally to any text type file.
CLEANTXT converts text files to standard FLEX, SK*DOS files.
COMMON compare two text files and reports differences.
COMPARE another check file that reports mis-matched lines.
CONCAT similar to FLEX, SK*DOS append but can also list files to screen.
DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.
ECHO echos to either screen or file.
FIND an improve find command with "pattern" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.

**NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:**
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F, S and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk) can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. Each directory looks to FLEX, **SK*DOS like a regular file, except they have the extension** '.DIR'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included.

*ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS , FLEX1.O, FLEX 2.O, or FLEX 9.O with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

F, S and CCF 5" - $50.00 F, S 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight task on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between task at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.

O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source≥ (either BASIC or A.L. Source Code).

F, S and CCF - $50.00

BASIC Utilities ONLY for UniFLEX -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

*FLEX, SK*DOS, CCF, OS-9, UniFLEX, 68000 & 6809h Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

X-TALK Complete (cable, 2 disks) $99.95

X-TALK Software (2 disks only) $69.95

X-TALK with CMODEM Source $149.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media

5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
U - $299.99

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.
* Now supplied as a two disk set:
Disk #1: JUST2.CMD object file,
*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*
Disk #2: JUSTSC object and source in C:
*FLEX, SK*DOS - OS9 - CC*
The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!
Disk (1) - PL9 FLEX only. F, S & CCF - $49.95
Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.
*Regular FLEX, SK*DOS $129.50*
** SPECIAL INTRODUCTION OFFER * $79.95*
SPECIAL PAT/JUST COMBO (w/source)
*FLEX, SK*DOS $99.95*
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00
Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.
*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.
*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!
*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.
F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.
NEW PRICES 6809 CCF and CCO - $99.95,
F, S or O - $179.95, U - $299.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

STYLO-SPELL from Great Plains Computer Co. – Fast Computer Dictionary. Complements Stylograph.
NEW PRICES 6809 CCF and CCO - $69.95,
F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. – Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.
NEW PRICES 6809 CCF and CCO - $59.95,
F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!
F, S or O - $329.95, U - $549.95
O, 68000 $695.00

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants – TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DYNACALC – Electronic Spread Sheet for the 6809 and 68000.
F, S, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
F, S - $59.95, U - $89.95

CROSS ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
*68000 or 6809, FLEX, SK*DOS, CCF, OS-9, UniFLEX*
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - w/source $500.00

XASM Cross Assemblers for FLEX, SK*DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's.
*Complete set, FLEX, SK*DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
SK*DOS	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O – Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.
*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

GAMES

RAPIER - 6809 Chess Program from S.E. Media – Requires FLEX, SK*DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
F, S and CCF - $79.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

68XX(X) And The STD BUS

68XX(X) on the STD BUS
Bill West BILL WEST INCORPORATED
174 Robert Treat Drive
Milford, Connecticut 06460
203 878-9376

The first order of business this month is to correct a technical error in last month's article. I inadvertently left the word "Treat" out of my street address, so if anyone sent me a letter at "174 Robert Drive" and had it returned or didn't get a response, that's the reason. By the way, Robert Treat was a governor of Connecticut in colonial times. There is also a "174 Robert Treat Parkway" here in Milford, so you can imagine my problems with the Post Office.

This month I am going to discuss various STD bus CPU cards that use Motorola processors. Although there are CPU cards using a number of different processors such as the 6801 and the 6802, I am only going to discuss those boards that use the 6809, 68008, or 68000. This article is based on information from manufacturer's data sheets and the "STD BUS Buyer's Guide" and should be accurate, but I suggest contacting the manufacturer if you have an interest in a particular board, especially about pricing. I'll discuss 6809-based CPUs first, and then the 68000 systems. There is also a "micro-review" of the Microplex M108 Dual Serial Interface for the SS30 bus at the end of the article.

Before describing specific boards, I will explain a few terms that I will use in the following paragraphs. In order to allow the user to configure the memory map of his system to take advantage of different memory and I/O configurations, many STD CPU boards use a fast bipolar PROM to generate chip selects for on-board I/O and memory and control signals such as MEMRQ* and IORQ* for off-board devices. Typically a 32x8 bit PROM is used. The operation is similar to the operation of the Dynamic Address Translators used in many 6809 systems. The high order address lines are used as address inputs to the PROM. The outputs of the PROM are used to selectively activate on-board memory, off-board memory (MEMRQ*), on-board I/O, or off-board I/O (IORQ*), depending on the current address output by the processor. The MEMEX and IOEXP lines may also be controlled by the mapping PROM. More sophisticated CPU cards include a hardware latch on-board to allow software selection of different banks of the PROM, allowing the memory map to be reconfigured under software control. This allows bank switching, or boot operation where a boot rom is switched out of the active memory map after booting an operating system from disk.

The IRQ vector latch is used to support Z80 peripherals that supply a vector to the CPU in response to an interrupt acknowledge cycle. During an interrupt acknowledge cycle, Z80-type peripheral chips will output a vector on the data bus (usually loaded into the peripheral at initialization by the CPU) that the processor can use to determine the appropriate interrupt service routine to execute. On 6809 cards, this data is captured in the interrupt vector register and then can be read by the CPU. Vectored interrupts are directly supported by 68000 processors.

The first processor board I will discuss is Micro-Aide Corporation's 6809-based 80-0033 CPU. Four 28-pin sockets allow the board to hold up to 128 Kbytes of RAM and EPROM in any combination. A battery-backed MM58274 calendar clock chip provides time-keeping functions. The battery may also be jumpered to back up selected memory sockets holding CMOS RAM. A 6551 ACIA is used to provide an RS232 serial port which may be configured as either DTE or DCE. The 6551 contains an internal software configurable baud rate generator which can generate standard baud rates. It has two more control lines than a 6850, but is similar in other respects. A bipolar mapping prom is used to determine the memory map. A 6522 VIA (Versatile Interface Adapter) is used to provide two eight-bit parallel I/O ports, along with two 16-bit counter timers. A vector capture register allows the use of vectored interrupts, and another software controlled register allows the card to support MEMEX, boot, and bank functions. (i.e. the MEMEX line can be activated under software control, and various blocks of memory can be enabled and disabled.) Full DMA to on-board memory is also supported. An inexpensive monitor ROM is available for testing and debugging. The price of the 80-0033 is $325. Two MHz and CMOS versions are available.

The 10809 CPU from Enterprise Systems Corporation includes a 6809 CPU, an RS232 port, and three 28-pin sockets. The serial port is implemented using a 6551 ACIA, which allows for the selection of baud rates under software control. The distinctive feature of this board is that it uses two 6522 VIAs to provide 32 parallel I/O lines plus eight handshake lines, and four 16-bit counter timers. The board also includes power-fail detect and interrupt generation circuitry. By connecting an off-board battery to the supplied connector, selected memory sockets can be protected in case of power failure. The price for this board is the 1 MHz version with no memory chips included is $250. A monitor ROM is available for $50.

Matrix Corporation is one of the few STD bus manufacturers that primarily uses Motorola processors in their products. Their STD bus products use 8-bit Motorola processors, and their single-height VME products use the 68000. The TS9 processor board is one of four 6809 CPU cards supplied by Matrix, and is actually a single-board computer with an STD form factor and bus interface. Many control-type applications can be handled by this board alone. In addition to the 6809 CPU, the board includes three 28-pin sockets that will support a 64 Kbyte EPROM, a 32 Kbyte RAM, and a NOVRAM (NOn-Volatile RAM). There are two serial ports which may be configured for RS232 or RS423 operation, along with a battery-backed clock and a watchdog timer. A 6522 VIA provides two eight-bit parallel ports, each with two handshake lines, and two sixteen-bit timers. There are also sixteen high-current (15ma source, 48ma sink) latched output drivers. This provides a total of 36 parallel I/O lines. There are three headers for parallel I/O connections on the board. One header is connected to the 16 high-current outputs and one is connected to the 20 VIA I/O lines. The third header includes the 16 latched outputs and eight inputs arranged for direct connection to an Opto 22 style 24 channel interface, which provides optically isolated connections to high-current, hi-voltage DC or AC equipment and controls. Circuitry to provide AC power fail detection is provided on the card to allow orderly shutdown and time/date logging in the case of a power failure. Software available from Matrix includes the TS9DEBUG ROM debugger/monitor, and OS9 in ROM. Disk-based development systems using OS9 and FLEX are also available. The price for the 1 MHz TS9 is $315. The TS9DEBUG monitor is $150, and the OS9 kernel in ROM (SCF only, no RBF) is $125.

The last 6809 CPU I will discuss is the only one capable of directly supporting OS9 Level II. The CPU/MMU from Systems Datar includes an on-board Memory Management Unit that allows mapping the 64 Kbyte logical address space of the 6809 into a 1 Megabyte physical address space on 4 Kbyte boundaries. The board is specifically designed to support OS9 Level II, and includes automatic task switching on interrupts, allowing user processes a full 64 Kbyte address space. The board also includes a real-time clock and a 28-pin socket for a boot ROM. The board is priced at $278.43 alone or $689.68 with OS9 Level II included.

The first of the 68008-based processors is one available from Allen Systems. It supports the original STD bus 8080 specification with 16-bit addressing plus MEMEX and IOEXP lines on the backplane. It includes two memory sockets, one for an 8 Kbyte EPROM, and one for a 2 or 8 Kbyte RAM. A 6551 ACIA is used to provide serial communications. The most interesting point about this CPU is that it is available as a bare PCB for $40. A monitor ROM is available for $40. Assembled and tested, the price for the board is $300.

The CPU-68K8 is a 68008-based processor card from XYZ Electronics, Inc. The board includes a 68008 processor running at 10MHz, three 28-pin memory sockets, and a battery-backed clock/calendar. An MC68901 Multi Function Peripheral chip is used to provide a serial port, an 8-bit parallel port, and two multi-mode timers. The board adheres to the STD-8088 specification to provide 20-bit addressing over the STD backplane. The price for the CPU card is $495. XYZ also supplies disk-based OS9 systems. Their minimum configuration includes the CPU-68K8, a 256 Kbyte static RAM card, a floppy-disk controller, and the OS9 Professional operating system with C compiler. The price for this system is $1800. Note that a card cage, power supply, and disk drives are not included in this price.

An interesting board from All-Control Systems, Inc. is the ACS hD100. This board does not conform to the normal STD form factor, but does include a complete STD bus interface with a standard edge connector. This means that although it can be plugged into an STD backplane, it will not fit in a standard STD card cage. The board includes a little bit of everything on it, so that may not be such a problem. The board uses a 68008 processor, and includes four 28-pin sockets which will hold 192 Kbytes of EPROM and 32 Kbytes of RAM. It includes a Signetics MC2671 Programmable Keyboard and Communications chip which allows connecting a keyboard matrix directly to the board, and also provides an RS232 serial port. An additional serial port is provided with a 6551 ACIA. A 6522 VIA is used to provide parallel I/O capabilities. Nine Opto 22 style I/O modules may be installed directly on the board, and five low-leakage AC solid state relays are included, along with two 20 KHz high-speed pulse counters. Analog I/O is supported with an eight channel 12-bit A/D convertor and a four channel 8-bit D/A convertor. Software available for the hD100 includes the RTOS real-time operating system, the MPEC Menu Presentation and Editing Control program, and the HDP Hardware Diagnostic Program. The list price for the hD100 is $950. If you are interested in a system using this board with OS9, contact me at BWI.

Datricon Corporation supplies both a 6809 and a 68008 CPU card for the STD bus. The ACS-68SBC is a 68008-based processor that includes four 28-pin memory sockets and an MM58274 battery-backed clock/calendar. Two serial ports are implemented using a 2681 DUART, which may be configured for RS232 or RS422 operation. The DUART also provides a 16-bit counter/timer. The full 1 Mbyte address space of the 68008 is supported on the STD backplane, with a memory-mapping PROM used to define on-board, off-board, and I/O address space. The board is available in both 8 and 10 MHz versions. The price for the 8 MHz board is $495. The MON-68 debug monitor ROM is available for $75.
The Datricon board is also available from BWI in a system configuration including the OS9 68K kernel in ROM. Included with the STD08R system is a 64 Kbyte EPROM containing the OS9 kernel and 64 Kbytes of static RAM. Over 100 Kbytes of user program space are available from the remainder of the kernel EPROM and the fourth memory socket. The price for the STD08R is $780.

Micro-Link Corporation provides the STD-202 CPU which uses the 68008 processor, and the STD-203 which uses the 68000. I have rather limited information on the

STD-203, but it is essential a 68000 with a boot EPROM on an STD board. The board allows 24-bit addressing and 16-bit data transfers over the STD bus, and supports Z80 type I/O cards. It is available in 4 and 8 MHz versions. The price for either the STD-202 or the STD-203 is $425. A diagnostic boot rom is available for each board. Micro-Link also supplies the STD-1000, a complete disk-based system which uses the PDOS operating system.

Another 68000 board is UC68K0, available from Micro-Craft. The board includes 512 KBytes of zero-wait state dynamic RAM and two 28-pin EPROM sockets. The STD interface is somewhat unusual in that off-board devices are addressed on word boundaries but only eight-bit data transfers are supported. This means that the bus interface is primarily suited for I/O devices (including RAM and ROM disks) as opposed to memory. Since there is 512 Kbytes of RAM on-board this is not a serious problem for many applications. On board I/O is supplied by an 8256 MUART, which includes one serial channel, 5 8-bit timers, and 16 parallel lines individually configurable as inputs or outputs. Also included with the UC68K0 is Micro-Craft's "Soft-Emulator" monitor and debugger, which includes features such as memory display, in-line assembly, breakpoints, and download routines. The SYM-68 symbolic interpreter is available separately and runs on an MS-DOS computer to provide symbolic debugging capabilities when used with the "Soft Emulator". The price for the UC68K0 is $779.

The MZ77860 is probably the the most powerful 68000-type CPU currently available for the STD bus. The board is available from Mizar Inc., which acquired the MDX line of STD products from Mostek. The board includes a 68010 CPU operating at 8 or 12 MHz, and up to 1 Megabyte of parity dynamic RAM mounted on board, along with two 28-pin sockets for up to 128 Kbytes of EPROM. The board is built using surface mount technology and SOICs (Small Outline Integrated Circuits), which have about the same relation to a normal 16-pin DIP as the 16-pin DIP has to a 40-pin DIP. Well, maybe a 28-pin, but anyway, SOICs are tiny! The board includes a logic sequencer which lets the 68010 access the STD bus like a 4 MHz or 6 MHz Z80. DMA devices on the STD bus can access the on-board memory 8-bits at a time. The dynamic RAM is mounted on four little cards that mount vertically on the main board, so different memory configurations are possible. Also included on the board is an MC6840 which provides 3 programmable timers. Software available from Mizar currently includes CP/M 68K. Contact them to see if other software has become available. The 8 MHz version of the board with 1 Mbyte of dynamic RAM included is $1292.

The Mizar board is also available in a system configuration from GW3. The system includes the CPU, a floppy disk controller, a SCSI controller, and a four channel serial interface card. The PDOS/68000 operating system from Eyring Research Institute is available for the system. If you are interested in using the Mizar board in an OS9 system, BWI will be supplying the Mizar board in an OS9 system in January of 1988. OS9 may also be available directly from Mizar.

That's it for this summary of Motorola CPUs available on the STD bus. Although not a complete list, this article should give you a good idea of the type of product that is available using 6809 or 68000 type processors. If you use 6802 processors or any of the Motorola microcontrollers, these products are also available. Next month's article will discuss an interface board we have designed at BWI. It allows a computer based on the GMX Micro-20, (or any computer supporting the Motorola I/O channel) to use the STD bus as an I/O expansion bus. In describing the interface and how it works, I will write about the I/O channel, some more details of STD timing, and discuss how we used PALs to interface these two different buses.

MICRO-REVIEW: M108

This has nothing to do with the STD bus, but may be of interest to SS50 bus users. Several months ago RFM Microplex Systems Ltd. (604 875-1461 Vancouver, BC, Canada) advertised their M108 SS30 Dual Serial Interface in 68Micro. The board includes a Signetics SCN 2681A Dual UART (that's Dual Universal Asynchronous Receiver Transmitter, or DUART), a crystal for the DUART's internal baud rate generator, plus buffers, 5-volt regulator, and RS232 drivers. I purchased two of the boards, along with the OS9 (6809) support software available, for my Gimix II OS9 Level II system, and am quite pleased with my purchase. The big advantage I find with these boards is that because the 2681 DUART has a 3-byte internal buffer for the receiver, (in addition to the serial shift register for assembling received characters) I am able to transfer files between my 68020 system and the Gimix box at 19.2 KBaud without losing characters. Using the Gimix serial cards, the highest data rate possible without lost characters was 2400 baud, even using XON - XOFF and saving the files to the ramdisk instead of the hard disk. The reason for this is that every tenth of a second, the real-time clock generates an interrupt and OS9 goes off and does some system stuff for a little while. Because the 6850 can buffer one character in addition to the one in the shift register, one might expect that the greatest improvement in speed possible would be a factor of three. This may actually be the case, because I think I changed my CPU speed from 1 to 2 MHz at the same time I installed the new cards. (I changed it from 2 to 1 for some forgotten reason.) The DUART provides a range of software selectable baud rates, and the ZMODE utility from Microplex allows the user to select the baud rate plus control a number of other features of the DUART. Since this is a "micro-review" let me sum up and say that the M108 is a quality product that provides a number of useful features to users of SS50 systems. (Note that Gimix is now GMX.)

+++

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01543

Tips for Beginners

At some time, everybody is a beginner; even an old-timer at BASIC or C needs some help when getting started with FORTH. One of the FORTH concepts easier for PASCAL programmers to grasp is that a FORTH program should be read from the last definition toward the first definition. This is because FORTH has the equivalent of a one-pass compiler, so that everything must be defined before it can be referenced.

As a general rule, the first words defined will be at the lowest level of the program and will be the most primitive. The definitions get more complex as you progress toward the "end" of the listing.

FORTH lends itself very easily to modular programming, and this is the way any long program should be written. This is true, whether one or a dozen programmers work on a program. Not only does it make it easier to keep track of the various operations of a program, but it makes it easier to reuse parts of one program within another so that you don't have to keep "reinventing the wheel".

Modular Programming in FORTH

This will be a few helpful hints, rather than a long technical dissertation on modular programming. I think that FORTH is the easiest language I have ever used for writing program modules. I find that one rarely needs global variables in the formal sense, because values are so easily passed on the Data Stack. If there are no global variables, indeed, not even parameter declaration for the stack, then there is no need to worry about declaring external variables nor concern yourself about a possible conflict of variable names between modules.

This latter concern for name conflict between modules can be ignored if descriptive names are chosen for definitions, variables, and constants. For example, never use TEMP as the name of a variable storage location, unless it refers to the temperature, and then don't use it for anything else. This is illustrated in the FILE-COPY module discussed later.

All of the FORTH implementations that I know of for the 68xx allow up to 31 significant characters in a name, so there is no need for short, cryptic names. By making a point of having names adequately descriptive of their purpose, you can be reasonably sure of preventing name conflicts, without taking any other measures. This also makes it easier to remember the names assigned to commonly used constants and variables.

If you have trouble thinking of a descriptive name, it is probably because you have not really thought out the purpose of your definition, etc. If you are really desperate for a name, you can always use I-CANNOT-THINK-OF-A-NAME-HERE and still not use up all of the alloted space!

The definition ?YES/NO

As an example, look at the definition ?YES/NO . This is a very simple case of a moderately primitive definition which can be used over and over as a part of many programs without ever needing to be changed. All it does is prompt for a YES or a NO answer which has been asked by another part of the program. The ? in the name indicates that it returns a boolean flag to the calling routine. Notice that once entered, the only way to escape from ?YES/NO is with the correcty typed answer, a "Y" or "N". A "y" or "n" are not acceptable. This forces you to think about your answer, and reduces the chance of answering Y when you intended N, and vice versa.

The FORTH word RECURSE is also known as MYSELF . It allows a definition to call itself, which is a form of recursion. If you don't have this word, then ?YES/NO can be rewritten to use a BEGIN...UNTIL loop which operates on a separately established loop-control flag. The BEGIN...UNTIL loop uses less Return Stack space than RECURSE , so recursion cannot always be used in a system with limited RAM. However, as an exercise, try writing ?YES/NO with the BEGIN...UNTIL construction and see what an advantage there is to programming ease and clarity with recursion.

Loading screen

It is all very well to say that a program should be written in a modular form, but another thing to give specific directions as to how to do it. If you are really interested in the subject, I suggest that you consult your local library for books on it. Here, I only want to point out some general ideas and give a simple example.

The example which occurs to me immediately is one that I am now working on. I have been asked by my church to write a data base program from which we can extract the usual lists, such as a mailing list of members, etc. But we also want to be able to extract lists of attendees in certain geographical areas who need help with transportation on Sunday morning. Obviously, there are other lists which we can use.

Since the church computer is a DEC VT180 running CP/M, I first considered using dBASE, which we have; all of this was given to us by a member when he "upgraded" his office to IBM. However, after reading the manual, I decided I wanted nothing to do with learning a new language with such a limited utility. My next choice was BASIC, since I assumed that any number of people could maintain the program, if I was not able to do so. Well, after a few hours of trying to keep track of the BASIC program, I gave it up as being likely to reactivate my angina. At last, sanity prevailed, and I admitted to myself that, if I was going to do the job right, it had better be done in FORTH!

I figured that I could write adequate documentation so that someone familiar with PASCAL, etc. could maintain the FORTH program easier than I could write the program in either PASCAL or C. This was based as much on the time wasted while program modifications were compiled, as the time saved in actually writing the program. Besides, the book STARTING FORTH by Leo Brodie has an example database program which was very nearly exactly what I wanted. I only needed to make a few changes and the additions required to make it easy for the church secretary to use in order to have a complete program ready to run.

To make a long story short, that is what I am doing. I will report later on some of the code I used to make the keyboard input nearly foolproof. I'll just say now that I have made extensive use of the looping technique illustrated in ?YES/NO , and the church secretary is very happy with the parts of the program she has seen so far.

Now, to my real reason for mentioning this program--How do you control the development of modules in a program this size? First of all, you break down the program into its major phases and components. The example labeled SCR # 2 shows the general way in which this kind of program can be "decomposed". The A-DATA-BASE example shows both a general outline of the program, and how each module can be conveniently loaded to make a coherent program. The whole program can be loaded by simply typing the command:

```
2 LOAD
```

This will cause each module to be loaded in the proper order, since the command A-DATA-BASE will be executed as part of the operation of loading screen #2.

Notice that the comments in the right-hand column of the SCR # 2 listing make a pretty good outline of the program. There really is not much else that needs to be said about a module during the initial planning of a project.

The left-hand column in the listing is a series of LOAD commands. Each module has been assigned a group of 10 screens, or blocks, but this was arbitrary and could be easily changed as appropriate.

I usually have a screen within a module call the succeeding screen by use of --> . Of course, additional loading screens could be buried within a module, but I don't like this because it would tend to hide the program flow during a cursory examination of the original loading screen. This would be bad, because I want the original loading screen to be a major part of the program documentation.

As I mentioned before, the entire program is loaded when screen #2 is loaded. This is particularly convenient during program development, since the command sequence:

```
FORGET A-DATA-BASE 2 LOAD
```

will clear the current version of the program from memory and load (and compile) the new version without additional commands.

Moving Screens

One advantage of modular programming is that you probably will never have to move more than about a half-dozen screens at one time in order to make room for a change or improvement in the program. Even with that, you will want a utility which will do the grunt-work for you. That is why I wrote the FILE-COPY utility.

FILE-COPY is a "smart" utility; it calculates whether you want to move up or down and starts copying from the proper place to make sure that you don't overwrite a screen before copying it. However, it is not fool-proof; it cannot keep you from copying the wrong screens or from moving them to the wrong place, so you still have to make some of the decisions.

You can call FILE-COPY with the following general command format:

```
<start-source#>    <start-destination#>
<count> FILE-COPY
```

For example, if you want to move 7 screens, starting at screen #31, up to screen #33, your command line would be:

```
31 33 7 FILE-COPY
```

As you can see, there is no problem with overlaping source and destination. To move from disk to disk, you might use the command:

```
712 1 50 FILE-COPY
```

which would move 50 screens (200 sectors) from the disk in drive #1 to the disk in drive #0 (assuming DSDD80 drives).

The Definition FILE-COPY

I think that a study of the listing for FILE-COPY will illustrate one of the points I was making about the natural derivation of names for words and variables. Since every character in the two variable names is significant, there is no problem with the program keeping them straight, even though each name is 14 characters long. Furthermore, I am not concerned with loading FILE-COPY with any other program, since I cannot imagine a case where I might encounter a name conflict. I have also used FILE- COPY in program modules without worrying about name conflict, and I have never had a problem with remembering the proper name for the variable.

I chose to use two variables in this utility instead of keeping them on the Data Stack, because I wanted a utility which could run under any version of FORTH. If I had tried to keep these numbers on the stack, I would have had to do some real programming gymnastics in order to make it run under fig-FORTH. The problem, here, is that fig-FORTH does not have any convenient way to access numbers buried very deeply on the stack. Also, there is a difference in the way PICK and ROLL work between FORTH-79 and FORTH-83, so I took the easy way out and used variables. As a programming exercise, you might try to write FILE-COPY without the variables; if you do, I would like to see your result.

(FILE-COPY)

The word (FILE-COPY) is written in parentheses in order to emphasize that it is used as a factor for other definitions. By defining (FILE-COPY) , I was able to shorten significantly the next two definitions. This is good practice, since it saves RAM and makes the whole program easier to read without adding much penalty to the running time.

(FILE-COPY) is the actual workhorse of the program. It is the definition which does the copying. The phrase DUP . simply duplicates and prints the number of the screen presently being copied; I like for a program to give some feedback so that I can tell when it is working and when it has locked up.

The phrase DUP FILE-COPY-VAR2 @ + calculates the number of the destination screen, and COPY does just that.

POS-FILE-COPY

This definition is written as a DO...LOOP . The definition requires that the number of the last+1 and the first screens be on the Data Stack when it is called. These are the loop limits, and conform to the FORTH practice of having the top limit + 1 as the first number and the starting value as the second number.

By having the loop limits the same as the actual screen numbers, then the loop index, I , can be called to specify the number of the screen to be copied. This is the number passed to (FILE-COPY) .

?BRK is a system-dependent primitive which checks the keyboard for input and returns a Boolean flag on the Data Stack. I used it here as a panic switch, since it checks the keyboard for input before completing the DO...LOOP . Pressing any key causes a TRUE to be left on the stack; otherwise, a FALSE is left. ?LEAVE is simply a primitive defined as IF LEAVE THEN , which causes an immediate exit from the loop if a TRUE was on the Data Stack.

NEG-FILE-COPY

The only real difference between POS-FILE-COPY and NEG-FILE-COPY is that the former copies screens starting at the lowest number, while the latter copies screens starting at the highest number. The phrase -1 +LOOP enables the loop to count backwards, one at a time.

FILE-COPY

FILE-COPY is the "smart" part of the utility, which is why it is such a long definition. It offends me to write such long definitions, but sometimes I just get trapped, and there is no easy way out! Fortunately, most of its length is really caused by the relatively long names used, so I guess that I will have give in, though not without a complaint.

The opening CR is just a sop to my esthetic(?) sense. It lets the line of screen numbers begin at the left margin, which I like to see, but it has no other value. Leave it out if you wish.

The >R temporarily saves the "count", the number of screens to be moved, on the Return Stack. This moves it out of the way for the next operations.

The 2DUP = duplicates the source and destination screen numbers and compares them. A TRUE flag is left on the Data Stack if they are equal; otherwise, FALSE is left.

IF tests this flag and executes the following 2DROP QUIT phrase if there was a TRUE . Execution continues if there was a FALSE . 2DROP clears the Data Stack and QUIT clears the Return Stack and aborts the execution of FILE-COPY .

The phrase OVER R> + FILE-COPY-VAR1 ! makes a copy of the source screen number, recovers the count from the Return Stack, and adds them together. The sum is then stored for later use.

Then 2DUP - duplicates the source and destination screen numbers and subtracts them. NEGATE changes the sign of the difference, and FILE-COPY-VAR2 ! saves the result. The change in sign is for the convenience of (FILE-COPY) .

Now that there is a commitment to move at least one screen and the two variables have been stocked, we are ready to determine the "direction" of movement. This is done by the next five lines in the definition. OVER < duplicates the source number and compares it to the number of the destination screen.

A TRUE left on the stack causes the IF segment to be executed. FILE-COPY-VAR1 @ SWAP causes the number+1 of the last screen to be fetched from storage to the Data Stack and then its position switched with the number of the starting screen. This puts them in the proper order for POS-FILE-COPY to copy, starting at the lowest screen number.

A FALSE left on the stack causes the ELSE segment to be executed. FILE-COPY-VARI @ 1- SWAP causes the number of the last screen to fetched from storage to the Data Stack and then its stack position switched with the number of the starting screen. This puts them in the proper order for NEG-FILE-COPY to copy, starting at the highest screen number.

The terminating FLUSH forces the disk buffer contents to be written to the disk. It is more insurance than anything else, since the last two screens could be lost if something happened to turn off the system before some other disk read/write took place. Better safe than sorry!

EOF

```
list c6-forth.lst

SCR # 77
( ?YES/NO return answer as flag, Y=TRUE/N=FALSE  RDL  01/11/86)

( This routine will keep cycling until either a "Y" or an "N" )
( has been entered.                                            )

: ?YES/NO       ( --- bool )
    ." (Y=yes/N=no)? "                    ( prompt              )
    KEY DUP DUP EMIT                      ( fetch input         )
      ASCII Y =                           ( was "Y" input?      )
        IF DROP TRUE                      ( yes                 )
        ELSE DUP ASCII N =                ( was "N" input?      )
          IF DROP FALSE                   ( yes                 )
          ELSE DROP CR RECURSE THEN THEN ;  ( a recursive loop )




SCR # 78
( Loading screen example                         RDL  10/19/86 )

: A-DATA-BASE ( -- )
    10 LOAD                     ( miscellaneous housekeeping     )
    20 LOAD                     ( CONSTANTS & VARIABLES          )
    30 LOAD                     ( data input                     )
    40 LOAD                     ( data verification              )
    50 LOAD                     ( data searching                 )
    60 LOAD                     ( data sorting                   )
    70 LOAD                     ( file updating                  )
    80 LOAD ;                   ( main program loop              )

CR A-DATA-BASE




SCR # 79
( ************************************************************ )
( FILE-COPY                                        RDL  08/13/86 )
( ************************************************************ )

VARIABLE FILE-COPY-VAR1                   ( temporary storage   )
VARIABLE FILE-COPY-VAR2                   ( differential        )


: (FILE-COPY) ( --- )
    DUP . DUP FILE-COPY-VAR2 @ + COPY ;

: POS-FILE-COPY    ( n1 n2 --- )          ( pos. loop increment  )
    DO I (FILE-COPY) ?BRK ?LEAVE LOOP ;

: NEG-FILE-COPY    ( n1 n2 --- )          ( neg. loop increment  )
    DO I (FILE-COPY) ?BRK ?LEAVE -1 +LOOP ;


SCR # 80
: FILE-COPY   ( source target count --- )
    CR
    >R                                    ( temp. save "count"     )
    2DUP =                                ( "source" = "target" ? )
      IF 2DROP QUIT THEN                  ( no need to copy        )
    OVER R> + FILE-COPY-VAR1 !            ( store "source" + "cnt")
    2DUP - NEGATE FILE-COPY-VAR2 !        ( store "src" - "target")
    OVER <                                ( look for overlapping   )
      IF FILE-COPY-VAR1 @ SWAP            ( set loop limits        )
        POS-FILE-COPY                     ( do copy operation      )
      ELSE FILE-COPY-VAR1 @ 1-            ( set loop limits        )
        NEG-FILE-COPY THEN                ( copy in reverse order  )
    FLUSH ;

( ************************************************************ )


EOF
```

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

Continued From Last Month

Build the GT-4 Graphic Terminal (A Construction Project)

By:
Joseph D. Condon
8072 172nd. Street W.
Lakeville, MN 55044
Phone: 612-431-7624

Part Two

Last month in part one of this article I introduced the GT (graphic terminal) construction project. The first part of this article dealt mainly with the physical design and construction of the GT. This month in part two of the article I will present the software that controls the GT. As you might have suspected, the software for the GT is quite lengthy and I would suggest that you purchase the reader service disk available from "68 Micro Journal" rather than keying in the entire source listing.

The Software

The software or firmware for the GT is best presented as two separate processes. The first being normal state processing and the second being interrupted state processing. The normal state processing or foreground process performs no physical I/O's with the keyboard or RS232 port. It simply calls routines to either place characters into or read characters out of circular buffers associated with these devices.

This type of design greatly reduces the complexity of the GT's firmware and allows for easy modification or enhancements. The source listing for the GT is well commented so I will try not to go into much detail in presenting it.

The Foreground Process

The foreground process is entered at power up time via the processors reset or restart vector. The process first clears some keyboard flags to achieve synchronization with the keyboards capitol lock, number lock and scroll lock indicators. Next the PIA control is configured and the value of the switch settings are read in and stored in the variable "switch". After passing by the warm entry label, the processors IRQ's are disabled and the systems stack is established. At power up, the 6809's IRQ mask will not allow interrupts to be serviced until its associated condition code register bit is cleared. Since we can enter this area of the firmware via the warm entry label, we must insure that interrupts are disabled.

Next we configure the ACIA control according to the switch settings that have been stored in the variable "switch". By referencing the variable "switch" instead of the actual swtich settings, we can easily change the ACIA's parameters by modifying the variable "switch" and then branching to the warm entry location.

After configuring the ACIA we then configure the video controller chip according to the values we equated at the beginning of the source file. These values are for a 12 mhz clock only. Before entering the main loop we initialize the majority of the systems global variables, set the absolute display address, clear the screen, beep the beeper and enable the processors interrupts.

The main loop basically gets the next character from the serial input buffer and if it is not a control character, displays it on the screen. If it is a control character, the main loop uses its value as an index into a jump table which points to the specific routine that handles that particular control function. After the character is displayed or the control function performed, the main loop is returned to via the RTS instruction. Each control function routine and the character display routine are well commented in the source listing and should require no further explanations.

All function routines in the GT firmware make use of a set of common subroutines which deal with basic functions such as clearing character locations and lines, scrolling the screen, manipulate the cursor, etc. Some of these subroutines such as the clear line routine, utilize repeating macro instructions. These instructions are merely an easy way of generating a long sequence of repetitive instructions with slight variations. These long sequences could easily be replaced with short loops however, due to the extremely critical speed requirements of these functions, I have chose to implement them as fast in-line instruction sequences.

There are several examples in the source listing of somewhat strange, unorthodox methods of accomplishing rather simple, straightforward tasks. Keep in mind that programming a bit mapped display terminal capable of operating at speeds up to 19200 baud is not a trivial task. Speed is of the essence not eloquently abbreviated or frugally concise algorithms. Brute force and repetition is often the only alternative in achieving fast and efficient code.

The Background Process

The background process is entered at the label "SysIrq" whenever an interrupt is requested by either the ACIA or the PIA controller. The interrupt handler first determines which device generated the request and then branches to the appropriate handler routine.

The serial interrupt handling routine checks the status of the ACIA to determine if the interrupt was generated because of a character being received or because of the completion of a character transmission. If a character has just been received, the handler transfers the character from the ACIA into a large circular receive character buffer which the foreground process reads from via the "GetChr" subroutine.

If for some reason this buffer is full, the routine will branch to an input buffer overflow error routine which displays an error message on the screen and then branches back to the terminals warm entry location. The input character buffer for the GT is very large, approximately seven thousand characters in length. When the buffer becomes approximately 90% full, the GT will send an XOFF character (hexadecimal $13) to the host computer. Afterwards, when the buffer becomes 20% empty, the GT will then send to the host an XON character (hexadecimal $11).

The GT's buffer will never overflow provided the host computer obeys the XON/XOFF protocol. About the only time the host system will be able to get ahead of the GT is when the GT is busy performing its line drawing routine. In order for the GT's buffer to become full, the host will have had to send out approximately 700 long line drawing commands in fast succession which is very unlikely. As mentioned above, it is very unlikely that you will ever see the input buffer overflow error appear on the GT's screen but because the possibility does exist, I felt it necessary to properly handle the situation.

If the serial interrupt was generated because of a character completing transmission, the interrupt routine will get the next character out of the serial output buffer and place it into the ACIA's transmit character register for transmission. If the output buffer is empty, the handler simply disables the ACIA's transmitter interrupts and returns to the foreground processing state.

The PIA interrupt handling routine is entered whenever a keyboard character is received. This routine reads the keyboard character, translates it and then inserts it into the serial output buffer. This buffer is fairly small because it is very difficult to type faster than the GT can send data to the host. If the GT is operating at 150 baud and if you keep mashing down on all of the keyboard keys, you may be able to cause an overflow condition. Also, since the GT will obey XON/XOFF protocol from the host and since the GT must see its

CTS signal true before sending data out, it is possible to overflow the buffer during normal operation. You probably will never experience this condition and if you do, it will usually indicate a problem with the host system or possibly the modem if you are connected remotely. Because of the possibility of overflow, the GT also has a keyboard overflow error routine that functions similarly to the input buffer overflow error routine.

Before the keyboard handler inserts the character into the serial output buffer, it must first translate the character into its proper ASCII code. The GT is the first project that I have built that utilizes an IBM XT type keyboard. The routines that I developed to handle the keyboard and translate the key codes are most easily described as being original. I am not trying to avoid the subject but I could easily write a whole article on just the keyboard handler and still not fully explain all of its functions and how they interrelate. Instead, I will just say that it works quite well and suggest that you study the source listing if you are interested in that sort of thing. Keep in mind that the IBM XT keyboard transmits a unique scan code for every key that is pressed, including the shift, shift lock and number lock keys. The keyboard also sends out a variation of the same scan code when a pressed key is released. Also, each and every key will repeat if held down long enough.

At the present time, I am not sure if "68 Micro Journal" plans on printing the entire source listing for the character generator file. I must admit that it is extremely long and very uninteresting. I will just say that the character generator is composed of 128 character images, each image occupying eight bytes of storage. The first 32 characters are abstract representations of what a control code might look like if it were displayed. There are certain instances when these characters may appear on the screen and they can be quite helpful in debuging application programs. The remaining 96 character images are standard ASCII character representations.

Hopefully my description of the GT's firmware has not been too brief. I firmly believe that a well commented source listing requires little or no explanation.

In conclusion, I would like to say that I have made a sincere effort intrying to present the GT construction project in a manner that most readers will approve of. Because the GT project consists of both hardware and software, I have tried to take a somewhat middle of the road approach in presenting it. Neither concentrating to deeply on the hardware or the software. I sincerely hope that anyone who decides to build the GT will not run into any serious problems but if you do, I will do my best to help you resolve them. Just send me a letter and I will respond as soon as possible. Even if you decide not to build the GT, I would appreciate any comments regarding this article, good or bad.

If you would like to see another construction project of this type in "68 Micro Journal", write to me or better yet, write to "68 Micro Journal" with your ideas or suggestions. Keep in mind, the project should be of general interest and noncommercial origin.

EOF

CRT Tables

Switch Pack Configurations Settings

```
S S S S S S S S  O-Open
7 6 5 4 3 2 1 0  C-Closed

      C C C  150   Baud
      C C O  300   Baud
      C O C  600   Baud
      C O O  1200  Baud
      O C C  2400  Baud
      O C O  4800  Baud
      O O C  9600  Baud
      O O O  19200 Baud
     C      8 Bit Word
     O      7 Bit Word
    C       1 Stop Bit
    O       2 Stop Bits
X X C          Parity Disabled
C C O          Odd Parity
C O O          Even Parity
O C O          Mark Parity
O O O          Space Parity
```

GT Control Functions

Hex	**ASCII**	**Function**
$00	(NUL)	do nothing
$01	(SOH)	display character
$02	(STX)	display character
$03	(ETX)	display character
$04	(EOT)	display character
$05	(ENQ)	display character
$06	(ACK)	display character
$07	(BEL)	beep the beeper
$08	(BS)	decrement cursor
$09	(HT)	display character
$0A	(LF)	line feed
$0B	(VT)	upline
$0C	(FF)	increment cursor
$0D	(CR)	carriage return
$0E	(SO)	display character
$0F	(SI)	display character
$10	(DLE)	display character
$11	(DC1)	resume transmission
$12	(DC2)	display character
$13	(DC3)	halt transmission
$14	(DC4)	display character

$15	(NAK)	display character
$16	(SYN)	display character
$17	(ETB)	display character
$18	(CAN)	display character
$19	(EM)	display character
$1A	(SUB)	clear screen
$1B	(ESC)	escape function
$1C	(FS)	display character
$1D	(GS)	display character
$1E	(RS)	home cursor
$1F	(US)	display character

GT Escape Code Functions

HEX	ASCII	Function
$3D	"="	load cursor (YX) *
$45	"E"	insert line
$47	"G"	graphic function
$51	"Q"	insert character
$52	"R"	delete line
$54	"T"	clear to end of line
$55	"U"	clear to end of page
$57	"W"	delete character
$65	"e"	cursor enable
$64	"d"	cursor disable
$6A	"j"	set hi-light
$6B	"k"	clear hi-light
$6C	"l"	set underscore
$6D	"m"	clear underscore

* Y and X coordinates must have $20
offset added to them.
Y = Y coordinate of char location.
Valid values are ($20-$37).
X = X coordinate of char location.
Valid values are ($20-$6F).

GT Graphic Functions

HEX	ASCII	Function
$55	"U"	up pen
$44	"D"	down pen
$53	"S"	set mode
$52	"R"	reset mode
$43	"C"	compliment mode
$54	"T"	test pixel *
$4D	"M"	move pen (XXXYYY) #
$49	"I"	image function

* Test pixel will return an ASCII STX code $02 followed by a "0" if pixel is reset, "1" if pixel is set or "?" if current x,y coordinates are outside the display window.

X and Y each represent ASCII
digits ($30-$39).
XXX = X coordinate of pixel.
Valid values are (000-639).
YYY = Y coordinate of pixel.
Valid values are (000-239).

GT Image Functions

Hex	ASCII	Function
$4C	"L"	load screen (XXXYYYCCC) *
$44	"D"	dump screen (XXXYYYCCC) *

* X, Y and C each represent ASCII
digits ($30-$39).
XXX = X coordinate of 4 bit nibble.
Valid values are (000-159).
YYY = Y coordinate of 4 bit nibble.
Valid values are (000-239).
CCC = Count of nibbles to transfer.
Valid values are (000-999).

A nibble, when sent should be or'ed with hex $20. When the GT dumps screen data to the host, it will or each nibble with $20 and precede the data with an ASCII STX code $02 to insure application program synchronization.

GT Function Key Codes

KEY	HX	ASCII
F1*	$30	"0"
F2	$31	"1"
F3	$32	"2"
F4	$33	"3"
F5	$34	"4"
F6	$35	"5"
F7	$36	"6"
F8	$37	"7"
F9	$38	"8"
F10	$39	"9"

* The F1 key when pressed as a control key will invoke the configuration menu.

All function key codes sent to the host are preceded by an ASCII DLE code ($10).

Non-Standard Keyboard Codes

KEY	HEX	ASCII
Scr Loc	*	
Alt	#	
Sys Req	#	
PrtSc	$2A	"*"
RTab	$09	(HT)
Home	$1E	(RS)
PgUp	$0A	(LF)
End	$0D	(CR)
PgDn	$0B	(VT)
Ins	$0D	(CR)
Del	$7F	(DEL)

* The scroll lock key will alternately send out the XOFF and XON codes ($13, $11).

No codes transmitted.

All other keys generate and transmit standard ASCII interpreted codes.

EOF

To be Continued Next Month

Bit-Bucket

By: All of us

"Contribute Nothing · Expect Nothing", DMW '86

LCD Driver with Serial Interface

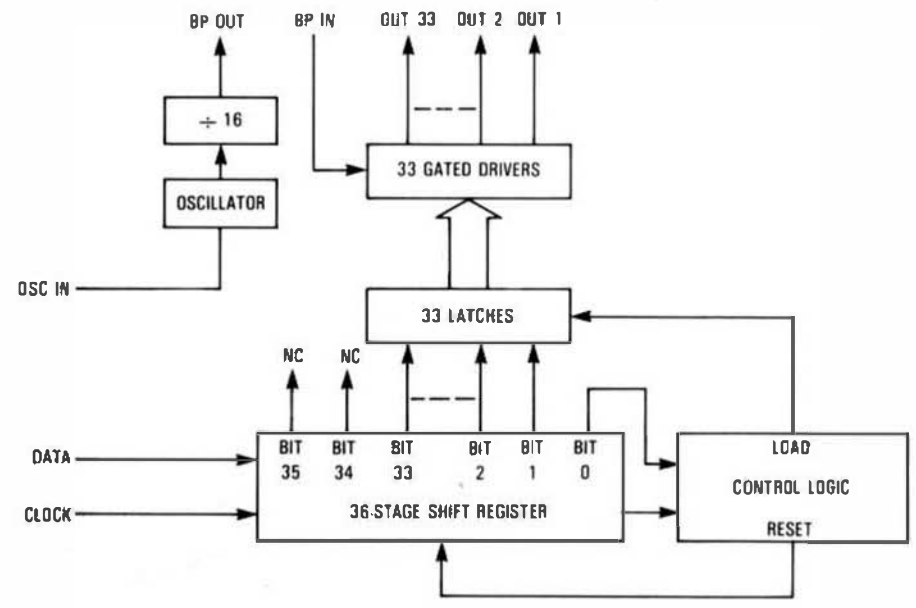
By: David Babin, P.E.
Motorola
MOS Digital-Analog IC Division

The MC145453 directly drives up to 33 non-multiplexed liquid-crystal elements per package. Both small and large displays can be driven by this CMOS IC. (A large 25mm 7-segment display has been successfully driven with the chip.) This flexible device allows external formatting of the display information, thus permitting use with alphanumeric, bar-graph, dot-matrix, or custom LCDs.

The IC's serial port only requires data and clock signals at CMOS or TTL levels. The double-buffered interface is realized with a 36-stage shift register which feeds 33 latches. The shift register is static, allowing data rates down to dc in a continuous or intermittent mode. The latches hold the display information, permitting an undisturbed display during shifting.

The usual data format is a start bit (high), followed by 33 display bits, plus 2 trailing bits (don't cares). If a display bit is high, the associated liquid-crystal element is activated. The start bit causes internal generation of a load signal which transfers the 33 display bits into the latches. After the load signal, an internal reset clears the shift register to ready the device for the next set of data.

Figure 1. Block Diagram



Alternately, a 5-byte (40-bit) transfer can be used as is encountered with SPI. As shown in Figure 2, 4 preamble bits (lows) precede the start bit (high). The start bit is followed by the 33 display bits plus 2 trailing bits. If desired, the bit stream can be formatted as in Figure 3 where the extra 4 bits (lows) follow the 2 trailing bits (don't cares).

Figure 2. 5-Byte Format

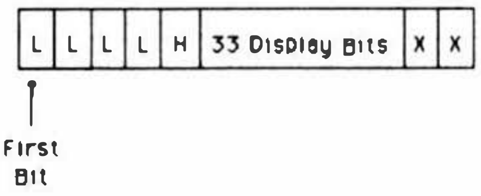
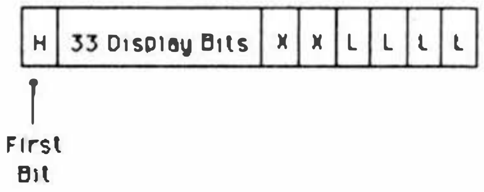


Figure 3. 5-Byte Format



The MC145453 must be initialized after power up. To initialize the IC, purge the shift register by clocking in at least 36 lows. The unit is then synchronized to the processor.

LCD elements respond to the differential RMS voltage between their frontplanes and backplanes. The non-muxed (direct-drive) system offers a much higher voltage contrast ratio (V_{RMS-ON} / $V_{RMS-OFF}$) than muxed types. The MC145453 has an excellent voltage contrast ratio of 9. (A muxed-by-2 driver drops this ratio to 2.24.) A high ratio maintains optimum display contrast while allowing the LCD threshold voltage (Figure 4) to have a loose tolerance. This threshold not only varies from LCD to LCD (batch to batch), but a particular LCD's threshold varies over temperature. Utilizing the MC145453, a wide operating temperature range is achieved without using temperature-compensating reference voltages, heaters for the LCD, or high-speed liquid crystals.

To relieve MPU/MCU overhead, an on-chip oscillator supports an external RC which generates the backplane and frontplane drive waveforms. Values of 1MΩ and 470pF produce a 60 to 150 Hz backplane frequency. The capacitor value can be increased up to 1200pF to reduce this frequency.

The MC145453's electrical characteristics are guaranteed over a supply voltage range of 3 to 10V and an ambient temperature range of -40° to 85°C. The operating supply current is only 40μA with all outputs open.

Two packages are presently available: a 40-pin DIP and a space-saving 44-lead PLCC (plastic leaded chip carrier).

To obtain a data sheet call 800/521-6274.

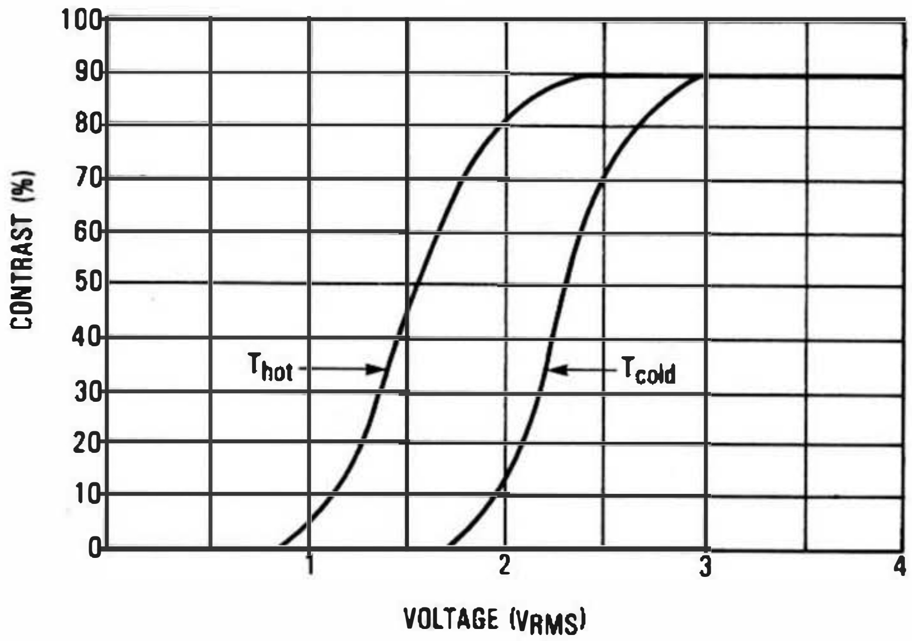


Figure 4. Temperature Effects on Contrast vs Voltage

Gespac, Inc

Larry Williams
'68 Micro Magazine
5900 Cassandra Smith Rd
Hixson, TN 37343

Dear Mr. Williams,

Please find enclosed a press kit concerning the release of our new and exciting short form catalog, a copy of which is enclosed for your personal review.

I am also pleased to report the remarkable growth of the G-64 bus business in the United States. A press kit highlighting the bus' success, as well as a list of G-64 bus manufacturers, is also enclosed for your review.

Please feel free to contact me personally if you have any questions or need more information.

Sincerely,

[illegible]

Cosma Pabouctsidis
President

CP/t
Enclosure

50 West Hoover Ave.
Mesa, Arizona 85202
Tel. (602) 962-5559
Fax. (602) 962-5750

Reader Contact: Mark Stephens
Editorial Contact: Cosma Pabouctsidis

FOR IMMEDIATE RELEASE

G-64 BUS GAINS FOOTHOLD IN U.S. MARKETPLACE

Mesa, AZ, October 31, 1987--GESPAC confirms the growing U.S. acceptance of the G-64 bus concept for board level products. The company began introducing the bus concept and products to the U.S. market in the latter part of 1984. This continuing success is well symbolized by the recent introduction of G-64 bus products from two additional U.S. companies.

Last month, Bubble Tec, a Northern California company specializing in bubble memory products, released a version of its bubble cartridge controller for the G-64 bus. In the same time frame, Avalon Engineering of Longmont, CO, developed a high performance servo motor controller board for the bus.

These companies have joined the ranks of three other North American Companies who have been offering G-64 bus products for the past few months. Sansoray, a company located in Fremont, California, has diversified its STD bus product line with the addition of a smart data acquisition board for the G-64 bus. Tommasi Technology of Montreal, Canada, produces EL panel displays with a controller that interfaces with the G-64 bus. Quadrant Scientific of Boulder, Colorado, has produced several data acquisitions boards specifically aimed at engine control.

GESPAC products have also enjoyed significant support from independent software companies. Eyring Research of Provo, UT, has helped port PDOS to the 68000/68020 family of processors. Palo Alto Shipping has ported its MACH-2 Forth compiler to the G-64 bus. Genesis Microsystems has ported its GENESCOPE/Target development software allowing easy development of software for the GESSBS-5 8088 single board system using an IBM PC.

More significantly however, the G-64 bus has been endorsed by over 200 companies in the U.S. in a wide variety of applications. Famous U.S. users include Allen Bradley, United Technologies, Westinghouse, Lockheed, General Motors, Pratt & Wittney, and NASA.

Only three years after its formal introduction, the G-64 bus has established solid roots in the U.S. marketplace. The high level of customer and third party vendor activities is the foundation to an even brighter future for the bus in the U.S.

The G-64 bus is an open architecture aimed at mid range industrial applications. The bus requires no licensing for its use. A detailed G-64 bus specification is available free of charge from GESPAC.

G-64 BOARD MANUFACTURERS

AA
chemin de la Salle
74000 Annecy [illegible]
France
Tel. (50) 877716
General program (+10 boards)

AUTOMATION
88, avenue de la Liberté
1640 Rhode St-Genèse
Belgique
Tel. (02) 3580879
Industrial I/O boards

AVALON ENGINEERING
PO Box 6350
Longmont, CO 80501
USA
Tel. (303) 441-9631
Motor controllers

BICC-VERO
Flanders Rd., Hedge end
Southampton, SO3 3LG
UK
Tel. (04215) 66300
Bus & Cabinet

BUBBLE-TEC
6805 Sierra Court
Dublin, CA 94568
USA
Tel. (415) 829-8701
Bubble memories

CENTRALP
21, rue Marcel Pagnol
69200 Venissieux/Lyon
France
Tel. (7) 8759230
Industrial Network Controller

CES
70, route du Pont-Butin
1213 Petit-Lancy
Switzerland
Tel. (022) 925745
Processor board 9995

CONTROL VISION Industries
44, chemin du Carriol
30380 St-Christol-les-Ales
France
Tel. (66) 528779
Linear CCD interface

C.S.A.E.
18, Via C. Goldoni
35131 Padova
Italy
Tel. (049) 64340
Industrial I/O boards

GEYLONICS
Industrielaan 8
8 10 Moorsele-Heule
Belgium
Tel. (056) 356557
General program (~20 boards)

ECRIN
2, rue des Murailles
38170 Seyssinet
France
Tel. (76) 210890
Industrial I/O Boards

ETIC
Z.I.R.S.T., chemin des Prés
38240 Meylan
France
Tel. (76) 410353
Modem board

ERMES
41, rue de Châtres
91790 Boissy-sous-St-Yon
France
Tel. (6) 0820821
General program

FUJITSU LIMITED
18, Mori BLDG. 3-13
Tokyo 105
Japan
Tel. 35020181/88
Bubble memory board

GEOMELEC
Chemin du Bulloz
74000 Annecy le Vieux
France
Tel. (50) 232812
Industrial I/O boards

GESPAC SA
3, chemin des Aulx
1228 Geneva
Switzerland
Tel. (022) 713400
General program (>100 boards)

GESPAC Inc.
50 West Hoover Ave
Mesa, AZ 85202
U.S.A.
Tel. (602) 962-5559
idem Gespac SA

HYTEC ELECTRONIC
Ladbroke Close - Woodley,
Reading Berkshire RG54DX
UK
Tel. (0734) 696973
Stepper Controller

KETEK Ltd.
Trojan Industrial Park
Paignton, Devon TQ47EP
UK
Tel. (0803) 557250
General program (~10 boards)

MAATEL
12, rue Ampère
38000 Grenoble
France
Tel. (76) 214291
Industrial I/O boards

MCE
79 81, avenue Louis Casaï
1216 Cointrin
Switzerland
Tel. (022) [illegible]
General program (~15 boards)

MII
Centre des 3 Fontaines
95021 Cergy Pontoise
France
Tel. (01) 30736223
General program (~10 boards)

MPL
Böhnirainstrasse 13
8800 Thalwil
Switzerland
Tel. (01) 7231225
General program (~10 boards)

NOVELEC
Z.I.R.S.T.
38240 Meylan
France
Tel. (76) 907045
Industrial I/O boards

PRONIX
64, rue des Bergers
75015 Paris
France
Tel. (1) 45575478
Industrial I/O boards

QUADRANT SCIENTIFIC
639 W. Dahua
Louisville, CO 80027
U.S.A.
Tel. (303) 666-8414
Data acquisition boards

QUIN SYSTEMS Ltd
35, Broad St., Wokingham,
Berks, RG111AV
UK
Tel. (0734) 783114
Data acquisition boards

SECAD
Le Balmay
01430 Maillat
France
Tel. (74) 757375
Industrial I/O boards

SENSORAY
44106 Old Warmsprings Blvd.
Fremont, CA 94538
U.S.A.
Tel. (415) 657-9331
Intelligent I/O boards

SYNTEL
Queens mill road
Huddersfield, HD 13 PG
UK
Tel. (0484) 535101
General program (~15 boards)

TROMBONE-OFICS
45, avenue de l'Europe
78140 Velizy
France
Tel. (3) 9469719
General program (~10 boards)

T.J.P. ELECTRONICS
3, West Street, Scarborough
N. Yorkshire
UK
Tel. (0723) 360928
General program (~10 boards)

TOMMASI TECH.
5725 Monkland Ave.
Montreal, Quebec H4A 1E7
Canada
Tel. (514) 482-4301
Graphic boards

WOTRE
Fröbelstrasse 4
4790 Paderborn
Germany
Tel. (05251) [illegible]
General program (~10 boards)

The above is a list of more than 30 companies who manufacture G-64 compatible boards. All information given is based on information we have at present. For more details please contact the supplier directly.

This list is published for information purposes only. GESPAC does not assume any responsibility for errors, and for products which are not produced by us, which consequently have not been tested or proven in our facility.

Gespac, Inc

Larry Williams
'68 Micro Magazine
5900 Cassandra Smith Rd
Hixson, TN 37343

Dear Mr. Williams,

I am pleased to announce the introduction of a versatile printer assembly that fits into G-64 bus hardware.

This new and exciting products allows the system integrator to add fast hard copy output in applications such as data loggers, point of sale terminals, and other similar applications.

I would be grateful if you could share this information with the readers of your publication. Enclosed is a press kit prepared to that end.

Please feel free to call me if you have any questions or need more information.

Sincerely,

[illegible]

Cosma Pabouctsidis
President

CP/t
Enclosure

PANEL MOUNT PRINTER ASSEMBLY FITS INTO G-64 HARDWARE

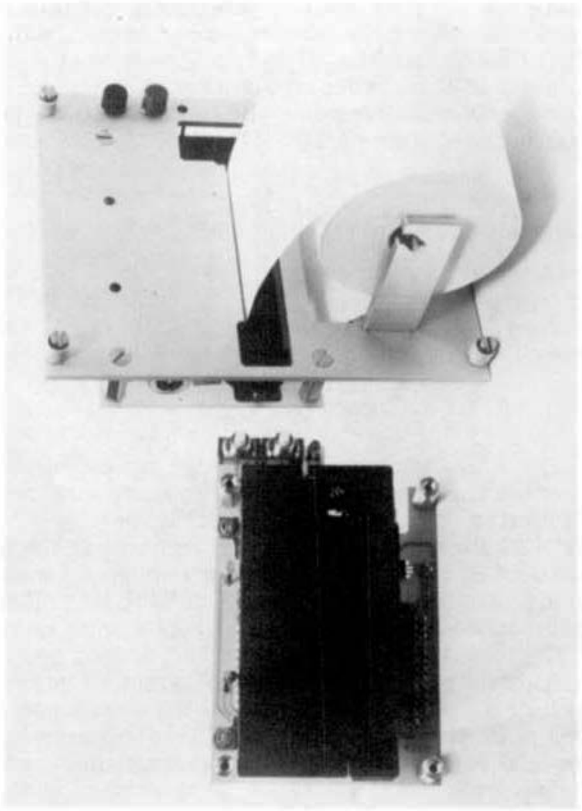
MESA, AZ, October 26, 1987--GESPAC announces a compact printer assembly that plugs directly into a standard G-64 bus backplane. The unit, manufactured by MPL AG, is sold and supported by GESPAC in the United States.

The unit is composed of a MPL 4342 controller card and the MPL 4343 printer assembly. The printer is a dot matrix impact printer using plain, 2 inch wide paper. The unit is very compact and can be used in applications such as data loggers, point of sale terminals, and other G-64 bus applications requiring fast and inexpensive hardcopy output.

The printer supports full ASCII character sets as well as eight user defined characters. Several options of the printer exist for character densities of 24, 30, 36 and 40 columns. The printer also supports a full graphics mode.

The printer interface is software compatible with GESPAC's standard printer interface and can therefore be put to immediate use thanks to the existing software drivers offered by GESPAC.

The printer assembly is available now for a unit price of $395 for the controller board, and $325 to $355 for the printer based on the required character density.



BILL WEST INCORPORATED
174 Robert Treat Drive
Milford, Connecticut 06460
203 878-9376

NEWS RELEASE

The STD020 is a 68020 single board computer with an STD I/O expansion bus, providing the advantages of a 32 bit CPU with the flexibility of the STD bus. The STD020 includes a 12 MHz 68020 processor with a 68881 floating point coprocessor, 2 Mbytes of RAM, four serial ports, one parallel port, battery backed up real-time clock, a 40 MByte hard disk, 40 Mbyte cartridge hard disk, and a 3 1/2" floppy drive. Also included is a 13 slot STD card cage, with one slot dedicated to the STD interface. (Except for the STD interface and card cage, STD boards in photo are for demo only and not included with system.) Operating system software is the OS9/68000 Professional operating system, a real-time multi-user multi-tasking operating system similar to UNIX. Complete development tools including the Microware C compiler, macro assembler, make, and microMACS editor are included.

The STD020 is available in various configurations, for both development and target applications. Options include higher speed CPUs, different disk configurations, rack mounting, and an internal STD card cage. Software support for custom configurations is available. An OEM version including only the SBC 68020 and the STD interface is available. Contact BWI for other configurations.
Release Date : December 1987 Price: $10569.00 disk-based system $ 3900.00 OEM system (no FPP)

BILL WEST INCORPORATED
174 Robert Treat Drive
Milford, Connecticut 06460
203 878-9376

NEWS RELEASE

The STD-ZS allows an STD bus card cage to be used as an intelligent I/O subsystem with any computer incorporating a SCSI port. The STD-ZS includes a Z80 processor and an NCR 5380 SCSI controller, along with sockets for up to 64 Kbytes of RAM and/or EPROM. The standard system includes a 16 Kbyte Eprom and a 2 Kbyte RAM with the required software for operating the board as a SCSI target. Options include a battery backed up real-time clock and a Z80 counter timer circuit (CTC). Programs for the Z80 may be in EPROM or downloaded from the host computer via the SCSI interface. Using an intelligent I/O subsystem allows the modularization of I/O tasks and the reduces the demands on the processing capabilities of the main processor.

By combining a STD-ZS with a computer such as an IBM PC, Apple Macintosh, or Atari ST, one obtains a flexible system which can use the wide range of I/O interfaces available for the STD bus and still maintain compatibility with the variety of off-the-shelf software available for these computers. Multiple STD-ZS may be daisy-chained to provide both additional I/O processing capabilities and a greater number of I/O ports. Some computers may require an additional host adapter board to meet the full SCSI specification. BWI provides software support for Macintosh, ST, IBM PC, and OS9 host systems. Contact BWI for information about other systems.
Release Date: August 1987 Price: $220.00 quantity one host adapter (if required) and host software additional.

BILL WEST INCORPORATED
174 Robert Treat Drive
Milford, Connecticut 06460
203 878-9376

NEWS RELEASE

The STD08R is a CPU board for the STD bus which includes a 68008 processor, two serial ports individually configurable for RS232 or RS422, a sixteen bit timer, a real-time clock with battery backup, four 28-pin sockets, and the OS9 68k operating system in ROM. The STD08R supports 20-bit extended addressing allowing the full utilization of the 1 Megabyte addressing space of the 68008. Sixteen bit I/O addressing is supported.

The four memory sockets are configured to accept two 64 Kbyte EPROMs and two 32 Kbyte static RAMs. The OS9 kernal occupies approximately 16 Kbytes of program memory space, leaving a total of over 100 Kbytes of program space on board for user applications. The board is supplied with 64 Kbytes of RAM and 64 Kbytes of EPROM installed, leaving one empty socket. A typical application will require the STD08R CPU card plus several I/O cards such as parallel interfaces or stepper motor drivers. The majority of STD bus I/O cards will work with the STD08R. BWI can provide assistance in selecting appropriate boards for particular applications, along with driver software.
Release date : June 1987 Price: $780 quantity one.
+++

MOTOROLA INC.

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-8598

EDITORIAL CONTACT:	INQUIRY RESPONSE:	READER CONTACT:
Bob King	Technical Info Center	Laura Tolpen
512/928-6141	P.O. Box 52073	512/440-2035
	Phoenix, AZ 85072	

MOTOROLA INTRODUCES THE MC68HC05L6, WITH LIQUID CRYSTAL DISPLAY DRIVER ON CHIP, ITS NEWEST MEMBER OF THE MC68HC05 FAMILY

Austin, Texas, october 9, 1987... The MC68HC05L6, the newest member of the MC68HC05 microcomputer family, has an on-chip liquid crystal display driver. This 8-bit single-chip microcomputer also contains an on-chip oscillator, CPU, 6208 bytes of ROM, 178 bytes of RAM, a Serial Peripheral Interface (Synchronous Communication - SPI), a 16-bit timer, and an audio tone generator. The LCD drive is 2/3, 1/3 VLL divider circuitry up to 96 segments.

Currently the MC68HC05L6 is available in die form only. There is a $5300 mask charge and a minimum order quantity of 1,000 units at $12.95. Packaged samples are available in a 68-lead PLCC. In mid 1988 the part will be packaged in a 68-lead Fine Pitch Plastic Leaded Chip Carrier.

Motorola's low cost development tool, the MC68HC05EVM, available for $500, supports the MC68HC05L6. Customers should call the EVM Hotline, (512) 440-EVMS, to request the MC68HC05L6 modification upon receipt of the board.
Motorola recommends the MC68HC05L6 for any handheld instrumentation utilizing LCD, such as bar code readers, pocket telephone directories, meters, cordless telephones, calculators, hand held cassette tape players, AM/FM radios, and infra red remote controls.

An advanced Information data book for the MC68HC05L6 (MC68HC05L6/D), is available and for more information contact your local Motorola Sales Office.

MOTOROLA ANNOUNCES THE AVAILABILITY OF THE MC68HC11 SINGLE-CHIP MICROCONTROLLER IN UNLIMITED QUANTITIES

Austin, Texas, October 18, 1987......The Motorola Microcontroller Division is pleased to announce the availability of the popular and powerful MC68HC11 single-chip design solution in unlimited quantities. The increased capacity has been made possible largely as the result of increased yields combined with the use of multiple fabrication facilities. The MC68HC11 is available for immediate booking and normal delivery.

The MC88HC11 comes equipped with many features not available on competing single-chip offerings. Included in this highlights list are the 2 serial communication ports, up to 512 BYTES of EEPROM, up to 8 KBYTES of ROM, 256 BYTES of RAM, an 8 channel analog-to-digital converter, and a variety of sophisticated timer subsystems.

Motorola has significantly reduced the price of the MC68HC11 microcontroller. To accommodate 2O88 volume orders, devices are available at a price under $10.00.
Advanced information can be obtained by requesting the MC68HC11A8/D, the technical data manual for the MC68HC11 Microcontroller and BR289, the supporting technical brochure.

MOTOROLA ANNOUNCES THE DISCONTINUATION OF THE MC267(X) SERIES OF VIDEO AND GRAPHIC PERIPHERAL DEVICES

Austin, Texas, October 19, 1987... The Motorola Microcontroller Division will be discontinuing the MC267(X) family of video and graphic support devices. Orders for life time buys will be available immediately with delivery dates not to exceed six months from the date of this release.

All of these devices are available in 40-pin dual-in-line packages and have an operating temperature range of 0-70 degrees. The family members are listed below:

MC2670 - Display Character and Graphic Generator (DCGG)
MC2671 - Programmable HKeyboard and Communications Controller (PKCC)
MC2672 - Programmable Video Timing Controler (PVTC)
MC2673 - Video Attributes Controller (VAC)
MC2674 - Advanced Video Display Controller (AVDC)
MC2675 - Color/Monochrome Attributes Controller (CMAC)

All of these devices are listed and detailed explanations given in the Motorola 8-bit Microprocessor and Peripheral Data manual (DL133). To obtain further details about these devices available for life time buys, please contact your local Motorola Sales office or the nearest authorized distributor.

AT&T CONTACT:	MOTOROLA CONTACT:	INQUIRY RESPONSE:
Daisy Ottman	Colleen Collins	Technical Info Center
201/221-8227	512/440-2123	P.O. Box 52073
		Phoenix, AZ 85072

AT&T AND MOTOROLA TEAM UP IN SUPPORT OF THE ISDN NETWORKING STANDARD

Austin, Texas, October 22, 1987.....The Motorola Inc., and AT&T announced today that AT&T plans to use a recently-announced Motorola chip to build data communications equipment. The device will help customers use high capacity integrated services digital network (ISDN) bandwidth efficiently.

Motorola is demonstrating the new chip, called the Motorola MC68606 Multi-Link (MLAPD) Protocol Controller, at the international Telecom '87 conference in Geneva. The device was designed to meet a request for development issued by AT&T and uses AT&T's set of specifications for connection between a host computer and terminals through a digital PBX.

The high performance chip will be used in equipment to transfer data at high speeds across networks connecting host computers and workstations, controllers and concentrators.
"We expect data switching to play an increasingly significant role in our equipment and networking capabilities," said Gordon Lewis, director of AT&T's system design laboratory. "Motorola has responded to our request to provide this device for the commercial market on a timely basis."

AT&T did not announce any specific future products that will incorporate the MC68606 MLAPD.

"In this development, it was fundamental for us to work with a partner who is a recognized participant in defining ISDN standards," said Murray Goldman, senior vice president and general manager of the Motorola Microprocessor Products Group. "Many of our customers who manufacture equipment see ISDN support at the primary rate as a cornerstone of what their customers will require in the future. We are proud that our technology base has enabled us to support this significant development for ISDN."

The primary rate interface connects customer equipment to an ISDN network over a 1.544 Mbps link.

The MLAPD chip reduces the workload of local processors, allows more data to be carried and accommodates more network links than do other products that implement the LAPD protocol.

Samples of the MC68606 MLAPD will be available from Motorola upon request beginning in December 1987. AT&T will also be able to manufacture the device for its own requirements as the market develops.

MOTOROLA
Microcomputer Division

2900 South Diablo Way, Tempe, AZ 85282
P.O. Box 20912, Phoenix, AZ 85036

READER CONTACT:
Motorola Microcomputer Division
Attn: MARCOM DEPT. — DW283
2900 S. Diablo Way
Tempe, AZ 85282

— or call —

Toll Free Marketing Hotline
1-800-556-1234, ext. 230
(In CA: 1-800-441-2345, ext. 230)

EDITORIAL CONTA T:
Gary E. Cato, Manager
Public Relations
(602) 438-3576

MOTOROLA OFFERS HIGH PERFORMANCE, MULTI-USER COMPUTER SYSTEMS RUNNING 3 AND 6 MIPS STARTING AT UNDER $8000

Las Vegas, Nevada, November 3, 1987 . . . Motorola's Microcomputer Division announced new systems for the growing VME Delta Series™ line. The VME Delta Series, first introduced in April of this year, used the MC68020 in the Model 2316 and the Model 2616. The line is expanding with high-end, higher performance systems using Motorola's new MC68030 32-bit MPU running at 25MHz.

"The real strength of our VME Delta Series line are evident with the addition of our newest systems," said Noel Lesniak, Systems Product Manager. "We have not only broken the 6 MIPS performance level with our 68030 COMDEX demo, we are offering superior price performance at 3 MIPS for under $8000. Also, our customers have the ability to upgrade from MC68020 based models to higher-end models based on new technologies and compatible software. Companies which choose to integrate VME Delta Series models for their particular application, will have the capability to interconnect the systems using DeltaONE™ which is based on Unix* System V's Remote File Sharing (RFS), TCP/IP and Ethernet. By providing higher levels of integration at the board and system levels, we have given our customers the advantage of greater speed and more open expansion slots in the same size package. OEMs and Systems Integrators should realize significant savings in moving sometimes extensive and relatively expensive applications software to faster or larger systems."

New High-Performance Rack-Mountable VME Delta Series Model 1132

The Model 1132, a new MC68020-based, rack-mountable VME Delta Series system, was engineered to meet the needs of customers requiring VME Delta Series performance and capabilities in a 12-slot rackmount enclosure. The new system is software and hardware compatible with other VME Delta Series systems and will be used both as a development workstation and as an OEM platform for industrial control, data acquisition and other related applications.

At a price ranging from $18,500 to $35,500, the model 1132 will initially be offered in 67MB and 181MB configurations, and will be available in January.

3 MIPS Performance At Under $8000 (OEM Quantity) VME Delta Series Model 2334

At 1.5 times the performance of the Model 2316, its six-slot sister, the Model 2334 houses a new, more highly integrated, MC68020-based processor module. This new system is an ideal solution for commercial, technical and networking applications requiring higher performance in a 6-slot VME card cage with floor mount enclosure. The 2334 will handle up to 10 users or asynchronous devices, has capacity for 67 or 181MB of disk storage with ST-506 or ESDI controller, respectively. There are 3 open slots for expansion.

The 2334 is initially offered in two factory configured packages, taking advantage of the different disk options available. It will sell for $9,500 to $16,500 and will begin shipping in January.

MC68030-based, 6 MIPS Multi-user Systems VME Delta Series MODEL 3641 And MODEL 3841

The Model 3641 is a high-end system targeted at the small departmental Technical Systems market. Equipped with a 25MHz MC68030, a 25MHz MC68882 Floating Point Coprocessor and 64Kbytes of fast cache memory, the 12-slot VMEbus system can haul an extensive payload. The system can host up to 32 megabytes of memory, enough to support up to 50 serial ports. And, while the model 3641 system chassis houses the same number of peripheral devices as did previous models (5 full height, 5.25" devices), the 3641 can manage two times as much internal hard disk storage, up to 1.3 Gigabytes.

The "top-of-the-line" Model 3841 uses a new 20-slot VMEbus chassis and cabinet, and hosts the 25MHz MC68020 processor with 64Kbytes of fast cache memory. Large applications will have the power needed with up to 48 Megabytes of memory to support up to 66 serial connections, and capability to house up to 1.6 Gigabytes of internal hard disk storage. The Model 3841 is a perfect fit into the large departmental technical systems market, providing enough horsepower and open VME slots to serve the OEM and Systems Integrator markets with a high-performance solution addressing computer intensive applications such as simulation.

The MC68030-based Model 3641 will be generally available by Mid '88, with evaluation units shipping earlier. Prices range from $31,500 to $74,500.

The MODEL 3841 "top-of-the-line" system will also be available by Mid '88, and sells for between $39,500 and $99,500.

UNIX, REAL-TIME AND OTHER SOFTWARE SUPPORT

All VME Delta Series systems are supported by SYSTEM V/68™, a derivative of AT&T Unix System V Release 3. SYSTEM V/68 provides a broad range of development and support capabilities including the support of Remote File Sharing (RFS) over an Ethernet LAN. At the time of release, a SYSTEM V/68 interface library will be available to provide access to many real-time kernels which adhere to the standard Real Time Executive Interface Definition (RTEID).

An important concern to many OEMs and Systems Integrators is systems networking capability. The VME Delta Series systems come with several local and wide area networking software packages based on industry standard protocols such as IEEE 802.3 (Ethernet), TCP/IP with RFS, SNA and BSC. Some options allow interconnection to IBM mainframes or to IBM PCs.

Motorola offers numerous languages and development support packages including 'C', Pascal, Fortran and others as well as full debug software support for MC68020 and MC68030 CPU boards used in the VME Delta line of systems. Other software is also available, such as database management packages and various office automation packages.

VME Delta models all support an optional internal modem which gives our OEMs the ability to support their customer or users from a central site with a complete set of diagnostic capabilities.

Prices indicated are in U.S. Dollars, for shipment in the U.S. only. For additional information, contact your local Authorized Motorola Sales Representative.

•••

MOTOROLA GIVES MicroMAP™ HOST SOFTWARE TO REAL-TIME SOFTWARE VENDORS

Dallas, TX, September 22, 1987 . . . Motorola's Microcomputer Division announced a new program to put portions of its MicroMAP protocol software into the hands of numerous third party real-time kernel vendors. The move will provide OEMs and Systems Integrators with real-time MAP options more closely aligned with existing application software interfaces.

The program has to date, been offered to seven third party real-time software companies who support M68000 family MPUs. It actually offers "host resident" MicroMAP source code to the participating vendors free of charge. Several vendors are already negotiating with Motorola for the right to port the software to their real-time systems. They are then free to redistribute the resulting product, under Motorola license, to their customers.

The MicroMAP Host Resident source code, normally sold for $10,000, is that portion of the seven layer MAP protocol which must directly communicate to the host operating system where user interface and file access is accomplished. For MAP, this includes portions of the APPLICATION LAYER, such as FTAM, Directory Services and other user and system oriented tasks. Once ported, the host resident code will communicate from the real-time host operating system through a common interface called the "Common Environment," to the MAP controller board where the other MAP layers reside.

"As far as I know, we are the only MAP software vendor who has been willing to give away source code," said Bob Franklin, Product planning manager of MAP TOP and OSI products at Motorola. "The third party vendors who have signed up for the program are delighted to participate. They now have a really legitimate reason to climb onto the MAP bandwagon. Most of them are not prepared to spend thousands of dollars to purchase MAP protocol software, and the resource investment to develop it on their own is prohibitive. Also, after analysis of the real time market, we found that the market is pretty well segmented. That is to say, the current real-time vendors all hold a relatively equal piece of the real-time market. System designers want the availability of MAP software that is supported under the real-time software they use. There is no advantage in 'singling-out' any one of the real-time solutions and Motorola simply cannot port MicroMAP to all the real-time products available, much less effectively support them. We came to the conclusion that it made good sense for us to allow each vendor to independently port MicroMAP to their operating system and more effectively take advantage of the features which distinguish their real-time product from others."

"Of course there are some restrictions associated with the agreement," continued Franklin. "For example, the vendor may not change any portion of the 'Common Environment.' This assures two things. First, that regardless of the port, the third party vendor host resident code will always communicate correctly to the board resident MicroMAP software. Second, it assures that the 'give-away' software will bring Motorola more business, since the Common Environment is designed to communicate to Motorola's MVME372 MAP Controller board and MicroMAP protocol software."

"Our ultimate goal is to put more options within the reach of MAP integrators and help broaden the MAP market. The program really sets everyone up in a win-win situation."

Motorola also noted that the MicroMAP Third Party program is still open to other real-time vendors who might be interested. Contact Franklin at (602) 438-3500 for further information about the program.

MicroMAP is a trademark of Motorola Inc.

###

MOTOROLA INTRODUCES A HIGH-PERFORMANCE TOP NETWORK CONTROLLER BOARD AND ACCOMPANYING PROTOCOL SOFTWARE

Dallas, TX, September 22, 1987 . . . Motorola's Microcomputer Division demonstrated its new Technical and Office Protocol (TOP) products, the MVME374 TOP controller board and the new MicroTOP™ protocol software product, at the MAP/TOP User Group meeting today. The VMEbus-based board, compatible with TOP physical connection requirements, hosts an Ethernet chip set, and supports the TOP seven layer protocol with fast, 32-bit data transfer using a 16 MHz MC68020 MPU.

The MVME374 also hosts 1 Megabyte of fast, 32 bits-wide shared DRAM for high-performance, on-board processor access. The shared memory with parity, provides one wait state access to RAM supporting the MPU, and zero wait state cycle access for Ethernet Lance chip use.

MVME374 TOP CONTROLLER FEATURES

— MC68020 16 MHz Microprocessor

— AM7990/AM7992B Ethernet (LANCE) chip set

— One Megabyte share memory (with parity)
 *One wait state accesses to RAM for MPU
 *Zero wait state accesses for LANCE chip

— One MC68901 multi-function peripheral controller

— Local bus arbitration between MC68020, AM7990, VMEbus and RAM refresh

— VMEbus interrupter, with a programmable level and vector

— Four JEDEC standard sockets for EPROM = 32K x 8, or 64K x 8, or 128K x 8, ROM = 128K x 8, or EEPROM = 32K x 8

— One asynchronous, serial debug port

— Bus requester with Release on local cycle mode (a subset of RWD)

— Status LED's for FAIL and Ethernet transceiver power

— Memory mapped slave interrupt

— 256 x 4 of EEPROM backed NVRAM for board specific variables

— A32 D16 and A24 D16 VMEbus master interface; A32 D16 and A24 D16 VMEbus slave interface

— Partial VMEbus system controller for stand-alone operation

TOP PROTOCOL SOFTWARE FEATURES

One of the original vendors involved with the early development of MAP, Motorola has been developing, evaluating, enhancing and selling OSI-based software since 1982. A natural extension of current OSI software offerings, the MicroTOP protocol software demonstrated today, was patterned after the ITI certified MicroMAP™ 2.1 protocol stack, one of the fastest and most efficient seven-layer products on the market. The current MicroTOP 1.0 product implementation differs from MicroMAP 2.1 in that it works above 802.3 Ethernet rather than 802.4 Token Bus. Motorola will migrate MicroTOP to future revisions as the specification matures. In fact, Motorola will demonstrate MAP 3.0 and TOP 3.0 networks at the Enterprise Networking Event (ENE'88) being held in Baltimore, in June of next year, and will have a certified MAP AND TOP 3.0 products later in the year.

Motorola sells OSI software in both object and 'C' language source code. MicroTOP object is compatible with Motorola's line of Unix* SYSTEM V/68™, VMEbus systems, and like its MAP counterpart, can be tailored by source customers to meet their own hardware and operating system software needs.

Full debug software support is available for the MC68020-based MVME374 TOP controller, and SYSTEM V/68 software drivers are provided with MicroTOP object software packages.

PRICE AND AVAILABILITY

Part #	Description	Price	Availability
MVME374	32-bit MC68020-based TOP controller board with Ethernet Controller Chip set, 1 Megabyte Dynamic RAM, Interface to 802.3 physical layer.	$ 1,795	Beta Site Dec. '87 Production March '88
MicroTOP 1.0 Object code	Layers 1 to 7. Includes drivers for SYSTEM V/68	$ 600	Beta Site Dec. '87 Production March '88
MicroTOP 1.0 Source code	'C' language source code of layer 1 to 7 protocol	$50,000	Beta Site Dec. '87 Production March '88

*Prices indicated are Quantity One, in U.S. Dollars, for shipment in the U.S. only. For additional information, contact your local Authorized Motorola Sales Representative.

MicroMAP, MicroTOP and SYSTEM V/68 are trademarks of Motorola Inc.
Unix is a Registered Trademark of AT&T.

###

KOWIN

KOWIN COMPUTER CORPORATION
Kowin Building
830 North Wilcox
Montebello, California 90640
Telephone: 213/721-5500

FOR IMMEDIATE RELEASE

CONTACT: [illegible] & [illegible] INC.
Christopher [illegible] or
[illegible] MacGregor
(213) 620-5711

THE KOWIN THREE — MULTI-USER COMPUTER SYSTEM

Product Fact Sheet and Specifications

PRODUCT:	The Kowin Three
DESCRIPTION:	Low cost, easy to use, multi-user, UNIX-based, business and office automation computer system, with built-in telephone and modem.
COMPONENTS:	Host Computer/Workstation Graphic Workstation
FEATURES:	Two 16/32-bit microprocessors (MC68000) One 32-bit microprocessor (MC68020) Multi-processor architecture Clock rate 10 MHz (68000) and 20 MHz (68020) UNIX V.3 O/S Optional math coprocessor Transparent Technology Multi-user Multi-tasking Built-in modem Buil -in telephone Compact footprint Low cost Simple, easy installation Includes office automation software Easy to learn and use System memory 4.5 MB and up
SOFTWARE:	Office Automation software includes: Word processing Phone directory/autodial Phone messaging Electronic mail Calendar Calculator Notepad On-line help On-line training, tutorial Other UNIX-based business and application software

CONNECTIVITY:	Mini-computers Mainframes IBM PCs and clones Apple Macintoshes
TECHNICAL SPECIFICATIONS:	32-bit architecture Two MC68000, one MC68020 microprocessors 4.5 MB RAM, expandable to 8 MB 40 MB internal hard disk Additional mass storage up to 1 GB Displays: Graphic and Host Computer Workstations 12-inch, bit-mapped display 650 by 400 pixels 64K bytes independent display Screen: 80 characters by 25 lines Keyboards: Graphic and Host Computer Workstation 101 full travel keys Typewriter character positioning Numeric keypad Dedicated cursor control keys 13 x 4 level programmable function keys 19 x 2 x 7 inches, 3 pounds
NETWORK FEATURES:	Host Computer/Workstation 4 RS-422 standard ports with multi-drop capability of 16 each Integrated telephone handset Bell 212A-compatible modem Data phone port 40-pin external mass storage port 3-RS-232 serial ports Graphic Workstations Standard/multi-drop dataport 2 RS-232 serial ports Integrated telephone handset
PROGRAMMING LANGUAGES:	C BASIC COBOL
DIAGNOSTICS:	Included
MEASUREMENTS:	Host Computer/Workstation: 19 x 12.5 x 12 inches, including display 22 pounds Graphics Workstation: 19 x 12.5 x 12 inches, including display 17 pounds
PRICE:	16 user system: about $1,900/user, suggested retail price A complete turn-key system - all hardware UNIX V.3 O/S and office automation software
CONTACT:	Mr. William Baugh Vice President, Sales KOWIN COMPUTER CORPORATION Kowin Building 830 North Wilcox Montebello, California 90640 (213) 721-5500 (800)44-KOWIN; in California (800) 22-KOWIN

• • • • •

Classifieds As Submitted - No Guarantees

AT&T 7300 UNIX PC, UNIX V OS, 1MB Memory, 20 MB Hard Disk, 5" Drive, Internal Modem, Mouse. Best Offer Gets It.

S+ System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $4800.

DAISY WHEEL PRINTERS
Qume Sprint 9 - $900
Qume Sprint 5 - $800.

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.

3 - Dual 8" drive enclosure with power supply. New in box. $125 each.

5 - Siemens 8" Disk Drive, $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $495.

TELETYPE Model 43 PRINTER - with serial (RS232) interface and full ASCII keyboard. $250 ready to run.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Paralell Port, MP-09CPU Card - $900 complete.

Tom (615) 842-4600 M-F 9AM to 5PM EST

•••

PT68K-IA (Same SBC as Mustang-08/A). 12 MHz 68008,768 K RAM. 2-80 Track,25M Hard Disk. $1500
Martin Bose (415) 351-7297

•••

GMX MICRO - 20 (16.67 MHZ) with MC 68881 and OS9/68020 professional pak $3500 or best offer.
John Bing 9-5 EST (301)428-8214

Clearbrook Software Group (604)853-9118

CSG IMS is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B+Tree index structures make **CSG IMS** the ideal tool for file-intensive applications.

CSG IMS for CoCo2/3 OS9 L1/2 (single user)	$169.95
CSG IMS for OS9 L2 or 68000(multi user)	$495.00
CSG IMS demo with manual	$30

MSF - MSDos File Manager for CoCo 3/OS9 Level 2
allows you to use MSDos disks directly under OS9.
Requires CoCo 3, OS9 L2, SDISK3 driver $45.00

SERINA - System Mode Debugger for OS9 L2
allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.
Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal $139.00

ERINA - Symbolic User Mode Debugger for OS9
lets you find bugs by displaying the machine state and instuctions being executed. Set break points, change memory, assemble and disassemble code.
Requires 80 column display, OS9 L1/2 $69.00

Shipping: N. America - $5, Overseas - $10
Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295
OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems*

Hundreds Sold at Suggested Retail:

~~**$199.00**~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compiler their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug it in BASIC and Then Compile it to a .CMD Binary File.

For a LIMITED time save over 65%... This sale will not be repeated after it's over! *

SALE SPECIAL:
$69.95

SPECIAL

Thank-You-Sale

Only From:

S.E. Media™
CPI
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

SK★DOS™

The Generic DOS™ for 68000 applications in

- ★ Industrial Control
- ★ Business Use
- ★ Educational Computing
- ★ Scientific Computing
- ★ Number Crunching
- ★ Dedicated Systems
- ★ Turnkey Systems
- ★ Data Collection
- ★ Single-board Computers
- ★ Bus oriented Computers
- ★ Graphics Workstations
- ★ One-of-a-kind Systems
- ★ Advanced Hobbyist Use

SK•DOS is a single-user disk operating system for computers using Motorola 32-bit CPUs such as the 68008, 68000, 68010, and 68020. It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes. Because SK•DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK•DOS comes with over 40 commands and system programs, including a 6809 emulator which allows 68K SK•DOS to run application programs and languages developed for 6809 SK•DOS and other systems. Assemblers, editors, and higher level language support are available from third party software vendors and through public domain software.

SK•DOS is available for single-copy or dealer sales, as well as OEM licensing. Single copies cost $125 (inquire as to available systems). Extremely attractive OEM licensing terms are also available. An optional Configuration Kit contains a detailed Configuration Manual and two disks of source code for system adaptation, including source code for a system monitor/debug ROM and other programs useful for adapting SK•DOS to new systems.

SK★DOS

is available from

STAR-K

SOFTWARE SYSTEMS CORPORATION

P.O. BOX 209 · MT. KISCO, NY 10549 · 914/241-0287

TELEX 6106016774

INDUSTRIAL PASCAL FOR 68000 AND 6809

PCSK is a package that generates code for a 68000 series processor running on a 68000 development system. It includes the compiler, assembler, linker, host debugger, target debugger, and screen editor, all integrated together and controlled by a menu driven shell program. Source code is included for the runtime library and many of the utilities. Most operating systems supported are OS-9/68000 (Microware), PDOS (Eyring Research), and VERSAdos (Motorola).

PXK9 is a package that generates code for a 6809 processor running on a 68000 development system. Includes all of the features of the PCSK package above, except for the host debugger. Host operating system is OS-9/68000.

I WANT IT, WHERE DO I GET IT?

For more information on either of these two products please contact Certified Software, South East Media, or one of our European Licensees.

OEM LICENSEES	DISTRIBUTORS
Gespac sa, 3, chemin des Aulx, CH-1228 Geneva / Plan-les-Ouales, Switz. TEL (022) 713400, TLX 429989	R.C.S. Microsystems Ltd., 141 Uxbridge Road, Hampton Hill, Middlesex, England. TEL 01-9792204, TLX 8951470.
PEP Elektronik Systeme GmbH, Am Klosterwald 4, D-8950 Kaufbeuren, West Germany. TEL (08341) 8974, TLX 541233.	Dr. Rudolf Keil GmbH, Porphystrasse 15, D-6905 Schriesheim, West Germany. TEL 062 03/6741, TLX 465025.
Eltec Elektronik GmbH, Galileo-Galilei-Strasse, 6500 Mainz 42, Postfach 65, West Germany. TEL (06131) 50031, TLX 4187273.	Elsoft AG, Zelgweg 12, CH-5405 Baden-Daettwil, Switzerland. TEL 056-833377, TLX 828275.
	Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany. TEL 02861-2147, TLX 813343.

CERTIFIED SOFTWARE CORPORATION
616 CAMINO CABALLO, NIPOMO, CA 93444
TEL: (805) 929-1395 TELEX: 467013
FAX: (805) 929-1395 (MID-8AM)

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS9 version also processes FLEX format object file under OS9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
68010 disassembler $100-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS 3/$100 ALL/$200
specify: 180x,6502,6801/11,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross-assemblers in C, with load/unload utilities
sources for additional $50 each, $100 for 3, $300 for all

DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809	$75-FLEX $85-OS9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position ind.	$50-FLEX $75-OS9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequencer only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS9,UNIFLEX,MS-DOS,UNIX,SKDOS
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $7.50-SSSD/SSDD/DSDD
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 680x0, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-4570 or 1717

We take orders at any time, but plan long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNIFLEX tm Technical Systems Consultants; OS/9 Microware;
COCO Tandy; MSDOS Microsoft; SKDOS Star Software)

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles **OS9 USER NOTES**

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs**. The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1, .C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
*** All Currency in U.S. Dollars**

Continually Updated In 68 Micro Journal Monthly

**Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343**

VISA

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

OS-9™ SOFTWARE

L1 UTILITY PAK—Contains all programs formerly in Filter kits 1 & 2, and Hacker's kit 1 plus several additional programs. Complete "wild card" file operations, copies, moves, sorts, del, MACGEN shell command language compiler, Disassembler, Disk sector edit utility, new and improved editions, approx. 40 programs, increases your productivity. Most programs applicable for both level I & II 6809 OS-9. **$49.95** ($51.95)

Call or send Self Addressed Stamped Envelope for catalog of software for color Computer OS-9 and other OS-9 systems.

BOLD prices are CoCo OS-9 format disk, other formats (in parenthesis) specify format. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

SS-50C
MEMORY LIQUIDATION SALE!
(While Supply Lasts)

1 MEGABYTE RAM BOARD

Full megabyte of ram with disable options to suit any SS-506809 system. High reliability, can replace static ram for fraction of the cost. $399 for 2 Mhz or $439 for 2.25 Mhz board assembled, tested and fully populated.

2 MEGABYTE RAM DISK BOARD

RD2 2 megabytes dedicated ram disk board for SS-50 systems. Four layer circuit board socketed for 2 Megabytes! Special sale price of $399.00 includes only 256k of ram installed (you add the rest), includes OS-9 level I and II drivers for Ram disk, (note: you can re-boot your system without losing ram-disk contents). (Add $6 shipping and insurance.)

Please call for answers to your technical questions concerning these products.

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223, (503) 244-8152
(For best service call between 9-11 am Pacific time.)

OS-9 is a trademark of Microwave and Motorola Inc.
MS-DOS is a trademark of Microsoft Inc.

DATA-COMP
SPECIAL
Heavy Duty Power Supplies

For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

5900 Cassandra Smith Rd., Hixson, Tn. 37343
Telephone 615 842-4600 Telex 510 600-6630

Make: Boschert
Size: 10.5 x 5 x 2.5 inches
including heavy mounting bracket and heatsink

Rating: in 110/220 volts ac (strap change) Out: 130 watts

```
Output:  +5v - 10 amps
        +12v - 4.0 amps
        +12v - 2.0 amps
        -12v - 0.5 amps
```

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery
SPECIAL: $59.95 each
2 or more $49.95 each
Add: $7.50 each S/H

Make: Boschert
Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

```
Outputs:  +5v - 8.0 amps
         +12v - 2.4 amps
         +12v - 2.4 amps
         +12v - 2.1 amps
         -12v - 0.4 amps
```

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery
SPECIAL: $49.95 each
2 or more $39.95 each
Add: $7.50 S/H each

5900 Cassandra Smith Rd., Hixson, Tn. 37343
Telephone 615 842-4600 Telex 510 600-6630

68000 68020 68010

68008 6809 6800

Write or phone for catalog.

AAA Chicago Computer Center
120 Chestnut Lane — Wheeling IL 60090
(312) 459-0450

Technical Consultation available most weekdays from 4 PM to 6 PM CST

Stop!

Get a 25 Mega-Byte Hard Disk practically FREE

Sale Ends

12/1/87

*This is exactly one cent more than the price of the same system - with two floppies - for one cent more you get one floppy and a 25 MegaByte Hard Disk with the faster CPU board, additional serial ports and improved clock! Includes Professional OS-9™ Version 2 and the $500.00 C Compiler!

Remember - When it's over, IT'S OVER!

We don't know how long this very, very low price can be maintained, don't miss it!

Data-Comp Div. - CPI

!!! Subscribe Now !!!
68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card #____________________ Exp. Date________

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $____________

Name ____________________________

Street ____________________________

City ________________ State ____ Zip ________

Country____________________________

My Computer Is: ______________

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

68 Micro Journal
5900 Cassa dra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600
Telex 510 600-6630

68 MICRO JOURNAL
Reader Service Disks

Disk- 1	Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2	Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime.
Disk- 3	Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
Disk- 4	Mailing Program, *Finddat, *Change, *Testdisk.
Disk- 5	*DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESION, **BLACKJAK, **BOWLING.
Disk- 6	**Purchase Order, Index (Disk file indx).
Disk- 7	Linking Loader, Rload, Harkness.
Disk- 8	Crtest, Lanpher (May 82).
Disk- 9	Datecopy, Diskfix9 (Aug 82).
Disk-10	Home Accounting (July 82).
Disk-11	Dissembler (June 84).
Disk-12	Modem68 (May 84).
Disk-13	*Initmf68, Testmf68, *Cleanup, *Dskalign, Help, Date.Txt.
Disk-14	*Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib
Disk-15	Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16	Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17	Match Utility, RATBAS, A Basic Preprocessor.
Disk-18	Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDC ODE, CMD.Txt (Sept. 85 Spray).
Disk-19	Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20	UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21	Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22	Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23	ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March1986.
Disk-24	68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25	KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26	Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27	ROTABIT.TXT, SUMSTEST.TXT,CONDATA.TXT, BADMEN.TXT.
Disk-28	CT-82 Emulator, bit mapped.
Disk-29	**Star Trek
Disk-30	Simple Winchester, Dec.'86 Green.
Disk-31	*** Read/Write MS/PC-DOS (SK*DOS)
Disk-32	Hier-UNIX Type upgrade - 68MJ 2/87

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

Specify 8" disk $19.50
5" disk $16.95

Add: S/H - $3.50
Overseas add: $4.50 surface - $7.00 Air Mail, USA Dollars

68 MICRO JOURNAL
PO Box 849
Hixson, TN 37343
615 842-4600 - Telex 510 600-6630

IF YOU NEED IT, WE'VE GOT IT! (OR WE'LL MAKE IT)

Yes, **PERIPHERAL TECHNOLOGY** still sells the FD-2 for SS-50 Bus Computers. And, if you don't need it, we sell other products from Single Board Computers to Systems which should fit your requirements. Custom Hardware Design is also available. Here's a small sample of what we offer:

FD-2 FLOPPY DISK CONTROLLER

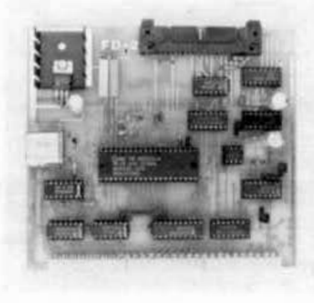
- Controls up to four 5¼" Drives
- Runs in 1 or 2 MHZ Systems
- Can be configured for either 4 or 16
- Addresses per I/O Slot (SS30 or SS30C)
- Uses WD2797 Controller Chip (compatible with 1771/179X Controller Chip)
- Hardware and Software compatible with SWTPC DC-4 controllers
- 6800/6809 Flex Drivers available
- SK*DOS Operating System or OS9/6809 Driver package

PT69-4 SINGLE BOARD COMPUTER

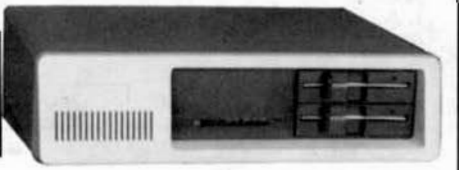
- 6809E Processor/1 MHZ Clock
- Four RS232 Serial Ports using 6850'S
- Two 8-Bit parallel Ports using 6821 PIA
- Time-Of-Day Clock (MC146818)
- 59K of user RAM
- 2K or 4K of EPROM using 2716 or 2732
- Double Sided/Double Density Floppy Controller
- Can Read/Write Radio Shack OS/9 Diskettes
- Board Size 5.6" X 8.2"

SYSTEMS

- Floppy or Winchester Versions Available
- Systems use any 6809 Single Board Computer or 68008 Board
- Will be configured to meet your Requirements
- OS9 & SK*DOS Operating Systems
- Call or write for system configurations



MONTHLY SPECIAL
FD-2 Floppy Disk Controller: **$129.00**
Regular Price : $149.95



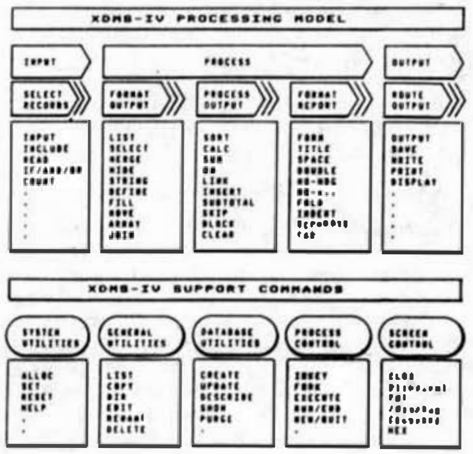
PERIPHERAL TECHNOLOGY
1480 Terrell Mill Road, Suite 870
Marietta, Georgia 30067
(404) 984-0742 Telex # 880584
VISA / MASTERCARD / CHECK / C.O.D.

*OS9 Is A Trademark of Microware & Motorola
*SWTPC Is A Trademark of Southwest Technical Systems

Catalogues Available Upon Request.

XDMS-IV

Data Management System



Save $100.00 - Limited Time
Regular $350.00 ~~- Now Only~~
$249.95

Technical telephone assistance: Tel 914-941-3552 (Evenings)
FLEX™ Tecnical Systems Consultants, SK*DOS™ STAR-KITS Corp.

FOR 6809 FLEX-SK-DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character filed names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced format Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...). The possibilities are unlimited...

Visa & Master Card Excepted

Telephone: 615-842-4601 or Telex: 510 600-6630
Or Write: S.E. Media, 5900 Cassandra Smith Rd., Hixson, Tenn. 37343

GMX MICRO-20 PRICE LIST

MICRO 20 (12.5 MHz) W/1 SAB	$2565.00
MICRO 20 (16.67 MHz) W/1 SAB	$2895.00
MICRO 20 (20 MHz) W/1 SAB	$3295.00

OPTIONAL PARTS AND ACCESSORIES

68881 12.5MHz Floating Point Coprocessor	$ 195.00
68881 16.67MHz Floating Point Coprocessor	$ 295.00
68881 20MHz Floating Point Coprocessor	$ 495.00
MOTOROLA 68020 USERS MANUAL	$ 18.00
MOTOROLA 68881 USERS MANUAL	$ 18.00

SBC ACCESSORY PACKAGE (M20-AP) $1690.00

The package includes a PC-style cabinet with a custom backpanel, a 25 Megabyte (unformatted) hard disk and controller, a floppy disk drive, a 150 watt power supply, cooling fan, panel mounted reset and abort switches, and all necessary internal cabling. (For use with SAB—9D serial connectors only.)

2nd 5"80 FLOPPY & CABLES FOR M20-AP, ADD	$ 250.00
SECOND 25MB HARD DISK & CABLES, ADD	$ 780.00
TO SUBSTITUTE 50MB HD FOR 25MB HD, ADD	$ 290.00
TO SUBSTITUTE 80MB HD FOR 25MB HD, ADD	$1500.00
TO SUBSTITUTE 155MB FOR 25MB HD, ADD	$2100.00
60MB TEAC STREAMER WITH ONE TAPE	$ 870.00
PKG. OF 5 TEAC TAPES	$ 112.50
CUSTOM BACK PANEL PLATE (BPP-PC)	$ 44.00

I/O EXPANSION BOARDS

16 PORT SERIAL BOARD ONLY (SBC-16S) .... $ 335.00

The SBC-16S extends the I/O capabilities of the GMX Micro-20 68020 Single-board Computer by adding sixteen asynchronous serial I/O ports. By using two SBC-16S boards, a total of thirty-six serial ports are possible.

RS232 ADAPTER (SAB-25, SAB-9D or SAB-8M) $165.00

The board provides level-shifting between TTL level and standard RS-232 signal levels for up to 4 serial I/O ports.

60 LINE PARALLEL I/O BOARD (SBC-60P) $398.00

The GMX SBC-60P uses three 68230 Parallel Interface/Timers (PI/Ts) to provide up to forty-eight parallel I/O lines. The I/O lines are buffered in six groups of eight lines each, with separate buffer direction control for each group. Buffer direction can be fixed by hardware jumpers, or can be software programmable for bidirectional applications.

PROTOTYPING BOARD (SBC-WW) $75.00

The SBC-WW provides a means of developing and testing custom I/O interface designs for the GMX Micro-20 68020 Single-board Computer. The board provides areas for both DIP (Dual Inline Package) and PGA (Pin Grid Array) devices, and a pre-wired memory area for up to 512K bytes of dynamic RAM.

I/O BUS ADAPTER (SBC-BA) $195.00

The SBC-BA provides an interface between the GMX Micro-20 68020 Single-board Computer and the Motorola Input/Output Channel (I/O bus). With the I/O bus, up to sixteen off-the-shelf or custom peripheral devices (I/O modules) can be connected to the GMX Micro-20.

ARCNET LAN board w/o Software (SBC-AN) $475.00

The SBC-AN provides an interface between the GMX Micro-20 68020 Single-board Computer and the ARCNET modified token-passing Local Area Network (LAN) originally developed by Datapoint Corp. The ARCNET is a baseband network with a data transmission rate of 2.5 Megabits/second. The standard transmission media is a single 93 ohm RG-62/U coaxial cable. Fiber optic versions are available as an option.

OS9 LAN Software Drivers for SBC-AN $120.00

GMX MICRO-20 SOFTWARE

020BUG UPDATE — PROMS & MANUAL $150.00

THESE 68020 OPERATING SYSTEMS ARE PRICED WHEN PURCHASED WITH THE MICRO-20. PLEASE ADD $150.00 IF PURCHASED LATER FOR THE UPDATED PROMS AND MANUALS. ALL SHIPPED STANDARD ON 5¼" DISKS. 3½" OPTIONAL IF SPECIFIED.

OS9/68020 PROFESSIONAL PAK $850.00

Includes O.S., "C", uMACS EDITOR, ASSEMBLER, DEBUGGER, development utilities, 68881 support.

OS9/68020 PERSONAL PAK .... $ 400.00

Personal OS-9 systems require a GMX Micro-20 development system running Professional OS-9/68020 for initial configuration.

Other Software for OS-9/68020

BASIC (Included in PERSONAL PAK)	$ 200.00
C COMPILER (Included in PROFESSIONAL PAK)	$ 750.00
PASCAL COMPILER	$ 500.00

UNIFLEX

UniFLEX	$ 450.00
UniFLEX WITH REAL-TIME ENHANCEMENTS	$ 800.00

Other Software for UniFLEX

UniFLEX BASIC W/PRECOMPILER	$ 300.00
UniFLEX C COMPILER	$ 350.00
UniFLEX COBOL COMPILER	$ 750.00
UniFLEX SCREEN EDITOR	$ 150.00
UniFLEX TEXT PROCESSOR	$ 200.00
UniFLEX SORT/MERGE PACKAGE	$ 200.00
UniFLEX VSAM MODULE	$ 100.00
UniFLEX UTILITIES PACKAGE I	$ 200.00
UniFLEX PARTIAL SOURCE LICENSE	$1000.00

GMX EXCLUSIVE VERSIONS, CUSTOMIZED FOR THE MICRO-20, OF THE BELOW LANGUAGES AND SOFTWARE ARE ALSO AVAILABLE FROM GMX.

ABSOFT FORTRAN (UniFLEX)	$1500.00
SCULPTOR (specify UniFLEX or OS9)	$ 995.00
FORTH (OS9)	$ 595.00
DYNACALC (specify UniFLEX or OS9)	$ 300.00

GMX DOES NOT GUARANTEE PERFORMANCE OF ANY GMX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

ALL PRICES ARE F.O.B. CHICAGO IN U.S. FUNDS

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

CONTACT GMX FOR MORE INFORMATION ON THE ABOVE PRODUCTS

GMX STILL SELLS GIMIX S50 BUS SYSTEMS, BOARDS & PARTS. CONTACT GMX FOR COMPLETE PRICE LIST.

GMX 1337 W. 37th Place, Chicago, IL 60609 (312) 927-5510 — TWX 910-221-4055 — FAX (312) 927-7352

Now! All the "Stars" Together!

For Ordering Call
(615)842-4600

FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

- + Read-Write-Dir RS Disk
- + Run RS Basic from Both
- + More Free Utilities
- + External Terminal Program
- + Test Disk Program
- + Disk Examine & Repair Program
- + Memory Examine Program
- + Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS, RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K
FULL 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a *speed whiz on disk I/O.* Fact is: the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks.

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

LOOK

```
                              Seconds   32 bit    Register
                                        Integer   Long
Other      68008   8   Mhz   OS-9   68K...18.0...9.0
MUSTANG-08   10   Mhz   OS-9   68K....9.8...6.3
              Main()
                 {
                        /* int i; */
C Benchmark Loop        register long i;
                        for (i=0; i < 999999; ++i);
                 }
```

Now even faster!
with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

25 Megabyte Hard Disk System

$1,998.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C com i er, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC   MUSTANG-08 is a trademark of CPI

Data-Comp Division

A Decade of Quality Service"
Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC hi-density FLEX 5" - Call for special Info.